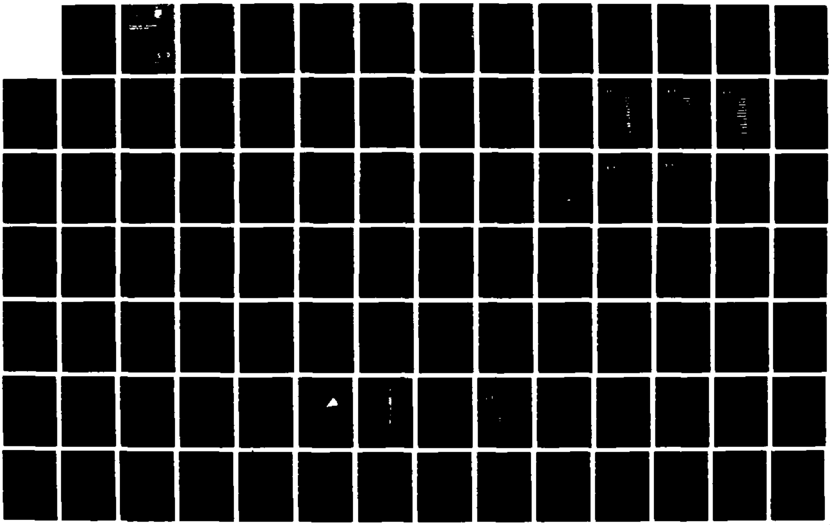
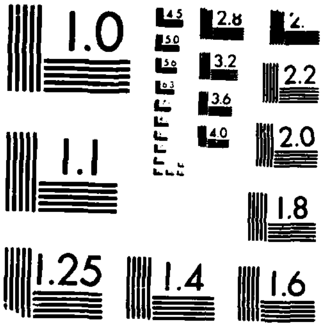
AD-A192 849 SOFTWARE QUALITY MEASUREMENT DEMONSTRATION PROJECT II 1/2
(U) SCIENCE APPLICATIONS INTERNATIONAL CORP SAN DIEGO
CA P PIERCE ET AL. OCT 87 RADC-TR-87-164
UNCLASSIFIED F30602-81-C-0132 F/G 12/5 NL





MICROCOPY RESOLUTION TEST CHART

BUREAU OF STANDARDS 1963 A

RADC-TR-87-164
Final Technical Report
October 1987

SOFTWARE QUALITY MEASUREMENT DEMONSTRATION PROJECT II

Science Applications International Corporation

Patricia Pierce, Richard Hartley and Suellen Worrells

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS
N/A

2a. SECURITY CLASSIFICATION AUTHORITY
N/A

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE
N/A

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release; distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
N/A

5. MONITORING ORGANIZATION REPORT NUMBER(S)
RADC-TR-87-164

6a. NAME OF PERFORMING ORGANIZATION
Science Applications International Corporation

6b. OFFICE SYMBOL (If applicable)

7a. NAME OF MONITORING ORGANIZATION
Rome Air Development Center (COEE)

6c. ADDRESS (City, State, and ZIP Code)
10260 Campus Point Drive
San Diego CA 92121

7b. ADDRESS (City, State, and ZIP Code)
Griffiss AFB NY 13441-5700

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
Rome Air Development Center

8b. OFFICE SYMBOL (If applicable)
COEE

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
F30602-85-C-0132

8c. ADDRESS (City, State, and ZIP Code)
Griffiss AFB NY 13441-5700

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
63728F	2527	03	13

11. TITLE (Include Security Classification)
SOFTWARE QUALITY MEASUREMENT DEMONSTRATION PROJFCT II

12. PERSONAL AUTHOR(S)
Patricia Pierce, Richard Hartley, Suellen Worrells

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM Jul 85 TO Apr 87

14. DATE OF REPORT (Year, Month, Day)
October 1987

15. PAGE COUNT
160

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
12	05	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Software Quality
Software Quality Metrics

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of this project was to assess the feasibility and utility of transferring the software quality specification and evaluation methodology from the Specification of Software Quality Attributes Guidebooks (3 Vols, RADC-TR-85-37, Feb 85) to the acquisition environment. Two decision aid developments were used as the test programs. Specific recommendations have been made to improve on the methodology, thereby enhancing its acceptance by and utility to software acquisition managers. A parallel study (Demo I) was conducted by another contractor under contract F30602-85-C-0180 using two different decision aids as the test vehicles.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Roger B. Panara

22b. TELEPHONE (Include Area Code)
(315) 330-3655

22c. OFFICE SYMBOL
RADC (COEE)

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

PREFACE

This document is the final technical report for the Software Quality Measurement Demonstration project, contract F30602-85-C-0132. Contract work was performed by Science Applications International Corporation for the Rome Air Development Center to provide an evaluation of software quality measurement guidelines.

The guidelines were proposed in a three-volume guidebook, Specification of Software Quality Attributes, produced by Boeing Aerospace Company under contract F30602-82-C-0137. The purpose of the guidebook was to develop a methodology to enable a software acquisition manager to determine and specify software quality factor requirements.

The final technical report consists of six sections, preceded by an executive summary:

- The Executive Summary is a top-level overview of the project and the final technical report.
- Section 1 describes the document's purpose. It gives an overview of the methodology used on the contract and of the project under analysis.
- Section 2 is an assessment of the measured quality of the software projects under examination.
- Section 3 includes detailed analysis and methodology evaluation results. These include the evaluation of trainee classes, quality goal specification, and the application of metric worksheets.
- Section 4 contains recommendations and conclusions drawn by SAIC based on analysis results.
- Section 5 lists all documentation referenced in this report.
- Section 6 lists all acronyms used in this report.

TABLE OF CONTENTS

SECTION PAGE

TABLE OF CONTENTS (Continued)

SECTION PAGE

APPENDIX

LIST OF FIGURES

FIGURE PAGE

LIST OF TABLES

EXECUTIVE SUMMARY

Contract Purpose

This project was conducted by Science Applications International Corporation for the Rome Air Development Center. The purpose of the project was to assess the feasibility and utility of transferring software measurement technology to the acquisition environment using software quality measurement guidebooks.

The guidebooks were written as a set of directions for the acquisition manager. They describe how the manager is to specify software quality goals, how he is to assess compliance based on evaluation reports, and how the evaluation reports are to be created. The reports are generally generated by independent verification and validation personnel, or by the developer, based on the evaluation of worksheets. The worksheets contain questions relating to each phase of the software development process and are applied to documents or to code.

The methodology chosen to accomplish this task was to try it out on two test software projects across all phases of development (requirements, preliminary design, detailed design, and coding). This included specifying software quality goals and evaluating documents and code written for each of the above phases.

Technical Approach

The approach taken for this project was to follow the guidebook procedures as closely as possible, while keeping detailed records on labor effort, quality evaluation results, and on any problems with the methodology recommended in the guidebooks. In addition, software problem reports were written against the documentation and code whenever metric violations were uncovered. This allowed SAIC to examine each part of the guidebook by actually performing each of the steps and procedures described.

Results of Software Quality Measurement Methodology Evaluation

As a result of this process, SAIC was able to gather data concerning the labor effort required to perform the software quality measurement process as described in the guidebooks. In addition, a full quality assessment was performed on each of the two software test projects under examination. Also gathered were data concerning the difficulty of implementing and using the steps and procedures described in the guidebook, and comments on the validity of the metric framework itself.

Conclusions

SAIC has identified both strengths and weaknesses in the software quality measurement methodology process.

The strengths of the guidebooks include:

- The metric framework itself is sound and reasonable. The quality scores determined for the test projects using this framework corresponded to the quality as assessed subjectively by project personnel.

- The guidebook methodology is divided into two volumes (one for quality specification by the acquisition manager, and one for quality evaluation by the developer or independent verification and validation personnel). This presentation is logical and helpful.

- Specific steps and procedures are included in the guidebook, and these are very helpful in understanding the entire measurement process. Some examples are given which are also helpful.

- A proposed Data Item Description that would allow reporting of quality evaluation results to the acquisition manager or System Program Office is useful.

In contrast, some weaknesses were also identified. These include:

- The guidebook presentation mixes theory, justification, examples, and procedures to be followed. This is often confusing.

- Information presented to allow the acquisition manager to understand the metric framework and quality specification is sketchy and confusing.

- Guidance supplied in specifying quality goals includes a weighting technique that will make quality assessment results each unique, and therefore not comparable across projects or applications.

- Metric element questions, used for evaluating document and software quality, are sometimes very ambiguous and confusing in themselves.

- No methodology was suggested to support the collection of the data used to answer metric element questions.

- No detailed guidance was supplied to support evaluators when problems or questions arise during metric worksheet scoring.

Recommended Changes

Based on following the steps and procedures contained in the software quality measurement guidebooks, SAIC has recommended some changes be incorporated into the guidebooks themselves. These changes include:

- Add more examples to the guidebooks, but separate them from the steps and procedures shown.

- Continue research so that the relationship between actual system quality and the predicted/measured system quality may be validated.

- Provide training material and/or classes for acquisition and evaluation personnel to help standardize the software quality measurement methodology, and to increase its acceptance and use in the Department of Defense as a whole.

- Provide more information to the acquisition manager concerning the relationships and structure of the metric framework, and describe how he might select subsets of this data to most effectively ensure that a quality system is developed, even if costs are restricted.

- Establish procedures for the handling of metric violations so that the information gathered from the evaluation process is most effectively used.

- Create workbooks to support the gathering of data needed for metric evaluation in as efficient and effective way as possible.

- Provide answer sheets for metric evaluation questions that are repeated for each unit in the system.

- Create glossaries, examples, and procedures for the use of the metric evaluators to ensure that the questions are interpreted and answered correctly.

More general recommendations are:

- In conjunction with the publication of Department of Defense pamphlets AFSCP 800-43, Air Force Systems Command Software Management Indicators, and AFSCP 800-14, Air Force Systems Command Software Quality Indicators, the Air Force Systems Command should sponsor the use of the software quality measurement (SQM) guidebooks on future system acquisitions.

- RADC should identify current on-going DOD acquisitions which are using all or part of the SQM guidebooks and capture data on the effectiveness of the guidebooks in these acquisitions.

- A research effort should be funded to make the changes recommended above to the guidebooks.

1.0 INTRODUCTION

This document is the final technical report produced for the Software Quality Measurement Demonstration project, performed for the Rome Air Development Center (RADC) by Science Applications International Corporation (SAIC). This section of the report describes its content and organization. It also presents an overview of the purpose, goals, and methodology used in the project.

1.1 Document Description

This report was produced for the Software Quality Measurement Demonstration (SQMD) project, conducted under contract F30602-85-C-0132. It contains a description of the project and includes analysis results, data collected, and recommendations made by SAIC.

The report is organized into the following sections:

- The Executive Summary is a top-level overview of the project and the final technical report.
- Section 1 describes the document's purpose and organization. It gives an overview of the methodology used on the contract and of the projects under analysis.
- Section 2 is an assessment of the measured quality of the software projects under examination.
- Section 3 includes the methodology evaluation results. These include the evaluation of trainee classes, quality goal specification, and the application of metric worksheets.
- Section 4 lists the recommendations and conclusions drawn by SAIC based on analysis results.
- Section 5 lists all documentation referenced in this report.
- Section 6 lists all acronyms used in this report.

1.2 Project Purpose

The intent of the SQMD project was to assess the feasibility and utility of transferring software measurement technology to the acquisition environment using software quality measurement guidebooks. These guidebooks consist of three volumes: Specification of Software Quality Attributes, Software Quality Specification Guidebook, and Software Quality Evaluation Guidebook.

Under contract F30602-85-C-0132, SAIC conducted an investigation into the application of software quality measurement (SQM) technology to two test projects, and evaluated the technology's utility as a quantitative input to software quality assurance. To accomplish this, the following tasks were required in the project statement of work:

- Develop a plan to implement this study based on the methodology contained in the guidebooks.
- Conduct independent software quality evaluation and validation to include goal specification, data collection, worksheet scoring, assessment of time logs, and measurement of goal achievement at review points.
- Evaluate the metric framework and methodology; the clarity and completeness of factors, criteria, and metrics; the ease of learning, understanding, and applying metric technology; the usability of the technology within the acquisition process; the soundness of its approach; the usability and effectiveness of the metric threshold and weighting approach; the appropriateness of the selected metrics; and the time required to perform the software quality measurement process.

1.3 Software Quality Measurement Methodology Overview

The procedures evaluated for this project are contained in the three-volume methodology guidebooks produced for RADC under contract F30602-82-C-0137. The guidebooks contain a methodology for the software acquisition manager to apply software quality specification and assessment to the software acquisition process. The guidebooks discuss the software metric framework, and knowledge of this framework is important to the understanding of this SAIC technical report. For details of the recommended approach, see the software quality measurement guidebooks listed as BOE-1, BOE-2, and BOE-3 in Section 5.0 of this report.

For reference purposes, the steps to be used in the software quality measurement (SQM) methodology are described briefly below. The method consists of two guidebooks. One guidebook describes how the acquisition manager can specify software quality goals and assess compliance to these goals. The second guidebook describes the software quality evaluation that is to be performed and reported to the acquisition manager.

SOFTWARE QUALITY SPECIFICATION. This part of the methodology involves procedures for specifying quality requirements. It includes methods and techniques for determining and specifying quality factor requirements, for making quantifiable tradeoffs among quality factors, for relating quality levels to cost over the software life cycle, and for analyzing quality measurement data.

The steps to be followed by the software acquisition manager as he specifies quality goals include:

- Select and specify quality factors. This procedure consists of identifying system functions, assigning quality factors and goals, considering factor interrelationships, and considering costs.

- Select and specify quality criteria. This procedure consists of selecting criteria, assigning weighting formulas, and considering interrelationships.

- Select and qualify metrics. This procedure consists of identifying metrics, selecting metric elements, and qualifying metric elements.

During the development of the system, the acquisition manager assesses quality compliance. Based on evaluation reports, he performs this process near the end of each software development phase, just prior to formal review. The purpose of the process is to assess compliance of development products with software quality factor requirements contained in the system specification and to predict the quality of the delivered software.

The steps involved in this process include reviewing requirement allocations and evaluation formulas, reviewing factor scores, reviewing criteria scores, and reviewing metric and metric element scores.

SOFTWARE QUALITY EVALUATION. Once the acquisition manager has specified software quality goals, the products of each life cycle phase are evaluated to determine if the system is meeting these goals. These measurements are generally gathered by the developer or by an independent verification and validation organization.

There are four steps identified for completing the quality evaluation process. These are:

- Identify allocation relationships. This procedure determines the relationship between the system-level functions and software elements to which the quality requirements have been allocated.

- Apply worksheets. The second step of the process is to collect metric data using worksheets.

- Score factors. During this step, scores are calculated for each quality factor using factor scoresheets and information from completed metric worksheets.

- Analyze scoring. Scoring results are used to determine variations from requirements and the causes of these variations. Corrective action is then recommended.

1.4 SAIC Technical Approach

SAIC's approach to the SQMD project was to apply the measurement technology as described in the guidebooks while maintaining extensive records on all aspects of the project. In addition, SAIC collected data on the training and experiences of both inexperienced and experienced team members concerning the application of software metric technology. This approach is illustrated in Figure 1.4-1.

SAIC's quantitative approach to data collection and evaluation was based on using forms, score sheets, worksheets, and analysis reports to document and record project data. This data included tasks accomplished, labor effort required, results, and any problems encountered during the measurement process.

Data was collected concerning any difficulties encountered while implementing the software quality measurement methodology using Methodology Problem Reports (MPRs), shown in Figure 1.4-2. Any metric violations were recorded on Technical Problem Reports (TPRs), shown in Figure 1.4-3. Project time logs were used to collect data to allow analysis of time required to perform project tasks. Figure 1.4-4 contains this log.

Project tasks were been decomposed into the following steps, each of which is discussed briefly below:

- Train inexperienced team members
- Specify quality requirements
- Evaluate compliance to quality requirements
- Assess compliance
- Analyze results

Training inexperienced team members was a formal instruction process involving classroom lectures and workbook exercises. The class was originally planned only for members of the project team who were not experienced with metric technology. However, more experienced team members expressed interest in the training and each decided to attend class. Section 3.2 discusses this training further. Two junior analysts joined the project after class completion. Their training was conducted informally by project leaders.

Specifying quality requirements was accomplished with a user questionnaire. The questionnaire concerned desirable quality goals, and was prepared and distributed to the developers of the test project decision aids and to SAIC project members. Based on these responses and the SQM methodology, goals for quality factors were determined, criteria weighted to calculate factor scores, and metric element questions

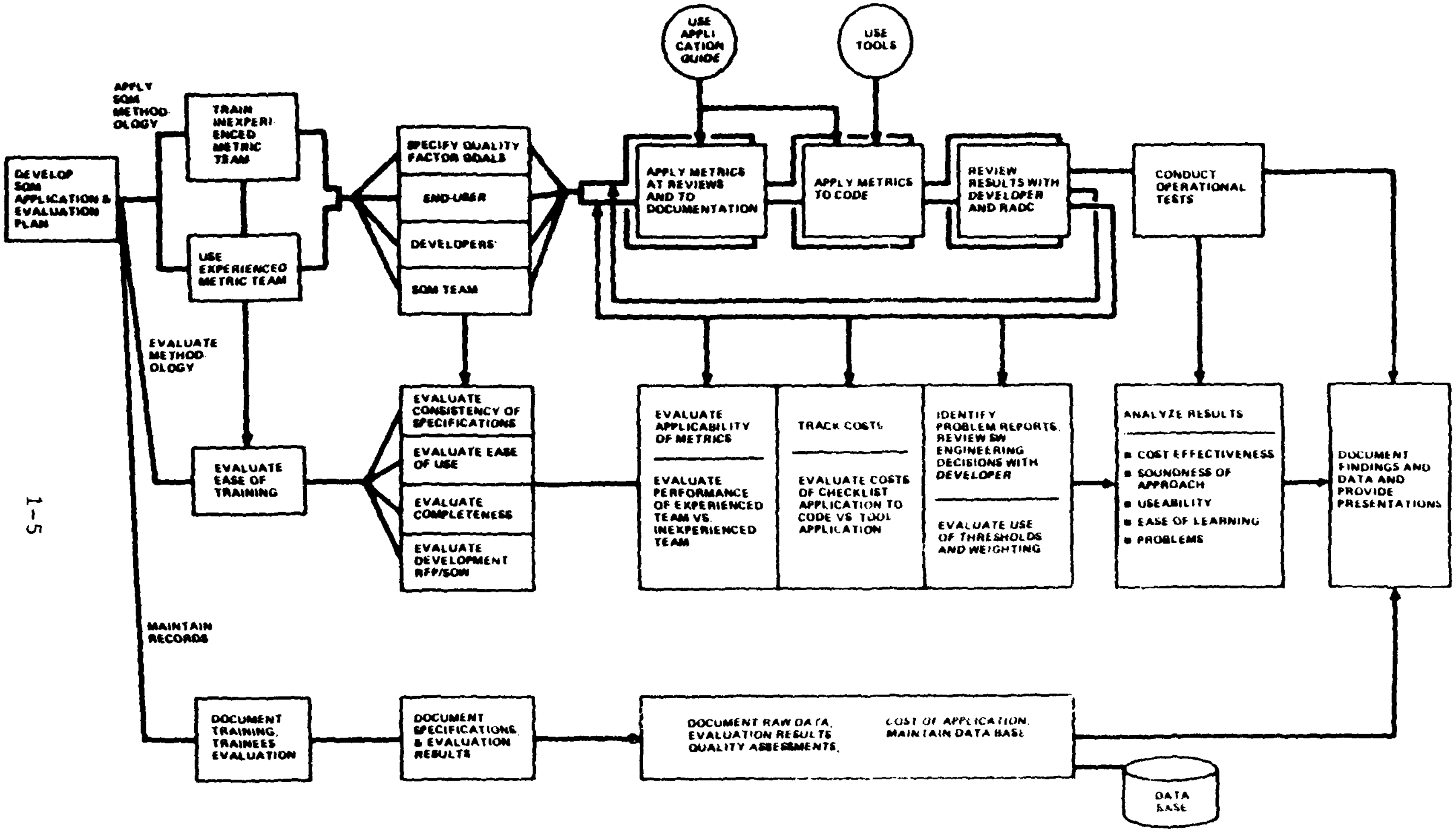
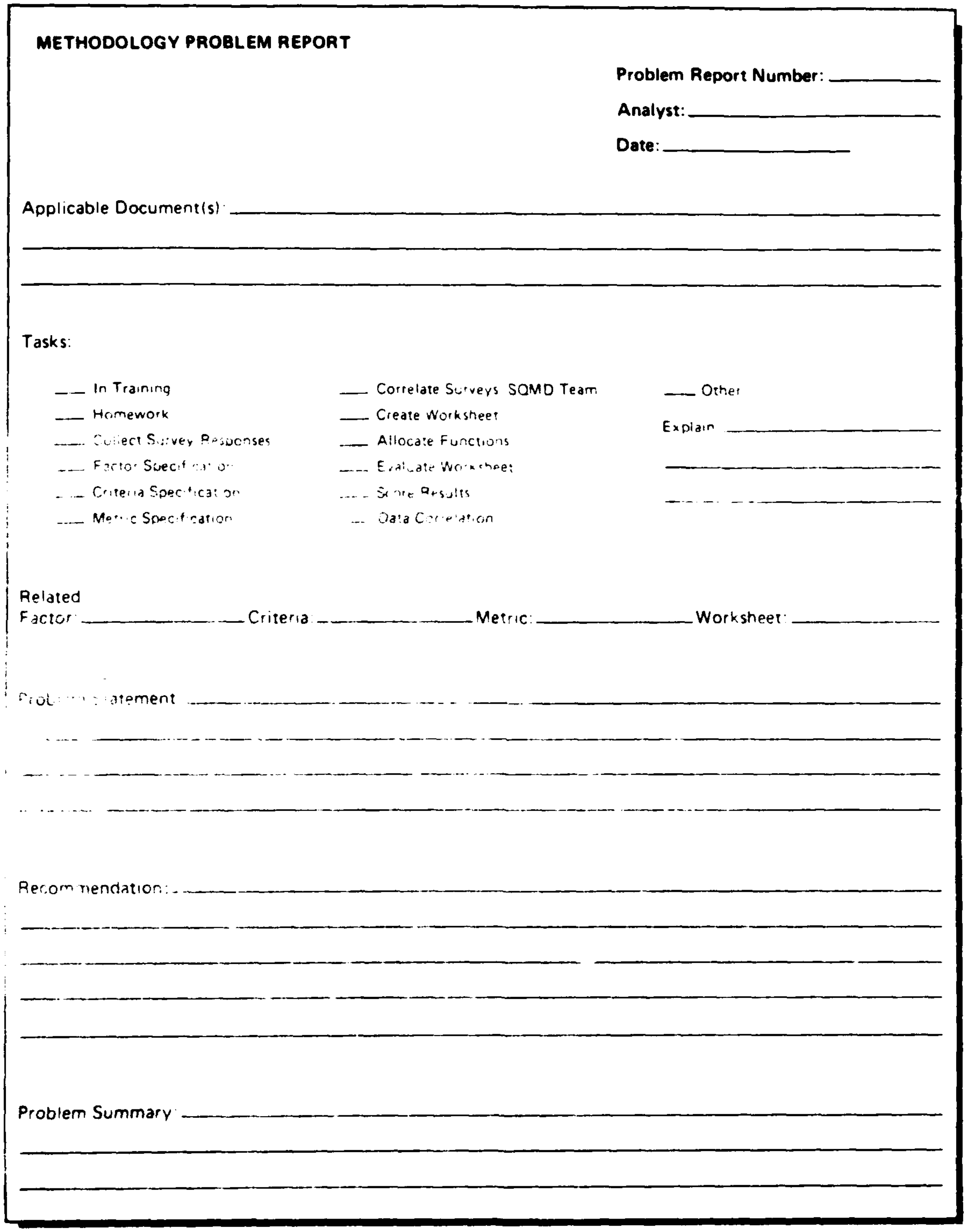

DEVELOP SQM APPLICATION & EVALUATION PLAN
APPLY SQM METHODOLOGY
TRAIN INEXPERIENCED METRIC TEAM
USE EXPERIENCED METRIC TEAM
EVALUATE METHODOLOGY
MAINTAIN RECORDS
SPECIFY QUALITY FACTOR GOALS
END-USER
DEVELOPERS'
SQM TEAM
USE APPLICATION GUIDE
USE TOOLS
APPLY METRICS AT REVIEWS AND TO DOCUMENTATION
APPLY METRICS TO CODE
REVIEW RESULTS WITH DEVELOPER AND RADC
CONDUCT OPERATIONAL TESTS
EVALUATE EASE OF TRAINING
EVALUATE CONSISTENCY OF SPECIFICATIONS
EVALUATE EASE OF USE
EVALUATE COMPLETENESS
EVALUATE DEVELOPMENT RFP/SOW
EVALUATE APPLICABILITY OF METRICS
EVALUATE PERFORMANCE OF EXPERIENCED TEAM VS. INEXPERIENCED TEAM
TRACK COSTS
EVALUATE COSTS OF CHECKLIST APPLICATION TO CODE VS TOOL APPLICATION
IDENTIFY PROBLEM REPORTS, REVIEW SW ENGINEERING DECISIONS WITH DEVELOPER
EVALUATE USE OF THRESHOLDS AND WEIGHTING
ANALYZE RESULTS
■ COST EFFECTIVENESS
■ SOUNDNESS OF APPROACH
■ USEABILITY
■ EASE OF LEARNING
■ PROBLEMS
DOCUMENT FINDINGS AND DATA AND PROVIDE PRESENTATIONS
DOCUMENT TRAINING, TRAINEES EVALUATION
DOCUMENT SPECIFICATIONS & EVALUATION RESULTS
DOCUMENT RAW DATA, EVALUATION RESULTS QUALITY ASSESSMENTS, COST OF APPLICATION, MAINTAIN DATA BASE
DATA BASE

FIGURE 1.4-1 OVERVIEW OF TECHNICAL APPROACH



METHODOLOGY PROBLEM REPORT

Problem Report Number: ____________

Analyst: ____________

Date: ____________

Applicable Document(s): ____________

Tasks:

___ In Training
___ Homework
___ Collect Survey Responses
___ Factor Specification
___ Criteria Specification
___ Metric Specification

___ Correlate Surveys SQMD Team
___ Create Worksheet
___ Allocate Functions
___ Evaluate Worksheet
___ Score Results
___ Data Correlation

___ Other
Explain ____________

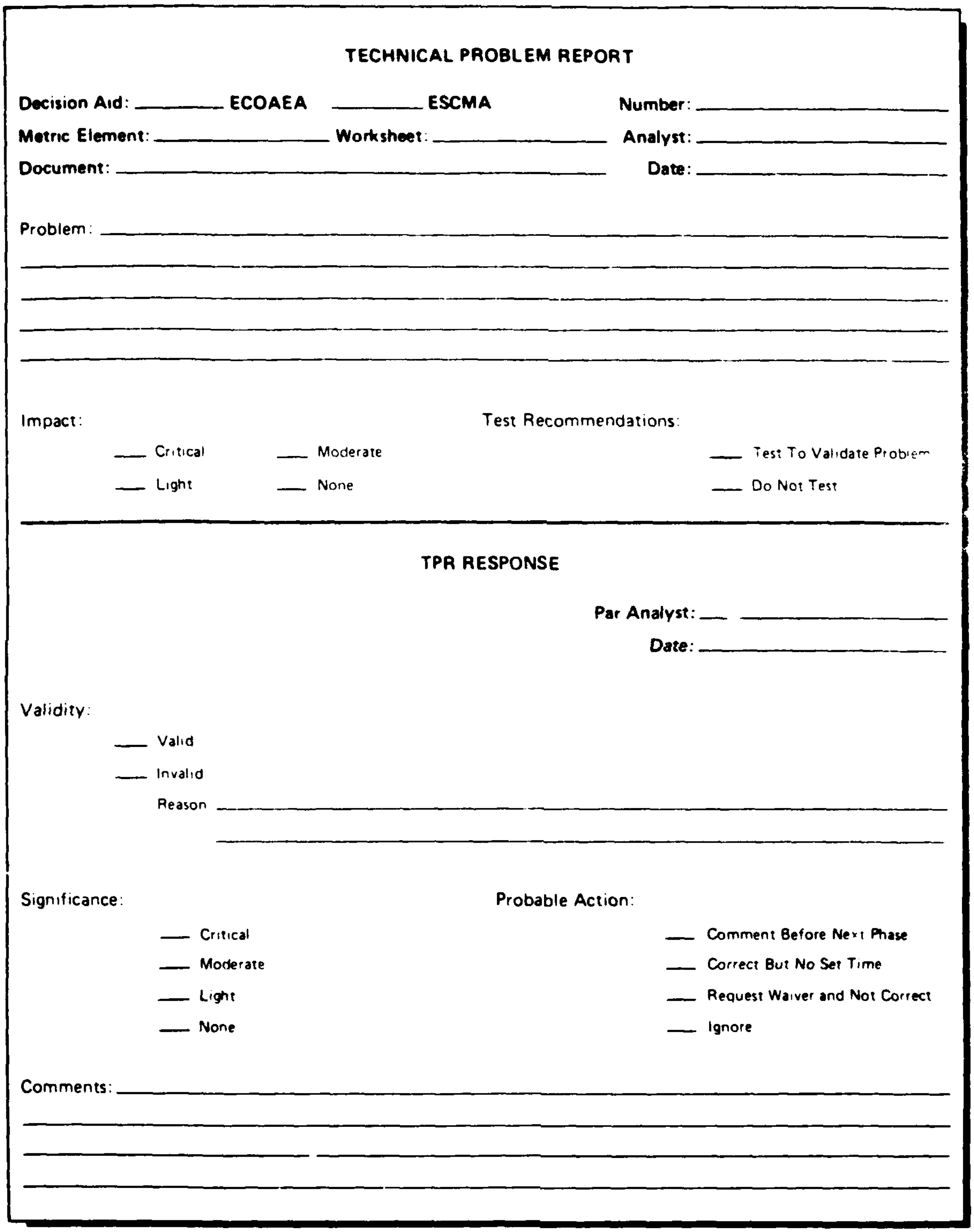
Related
Factor: ____________ Criteria: ____________ Metric: ____________ Worksheet: ____________

Problem Statement ____________

Recommendation: ____________

Problem Summary: ____________

FIGURE 1.4-2 METHODOLOGY PROBLEM REPORT



TECHNICAL PROBLEM REPORT

Decision Aid: ________ **ECOAEA** ________ **ESCMA** **Number:** ____________________

Metric Element: ________________ **Worksheet:** ________________ **Analyst:** ____________________

Document: ______________________________________ **Date:** ____________________

Problem: ____

____

____

____

____

Impact:

___ Critical ___ Moderate

___ Light ___ None

Test Recommendations:

___ Test To Validate Problem

___ Do Not Test

TPR RESPONSE

Par Analyst: ___ ________________

Date: ____________________

Validity:

___ Valid

___ Invalid

Reason ____

____

Significance:

___ Critical

___ Moderate

___ Light

___ None

Probable Action:

___ Comment Before Next Phase

___ Correct But No Set Time

___ Request Waiver and Not Correct

___ Ignore

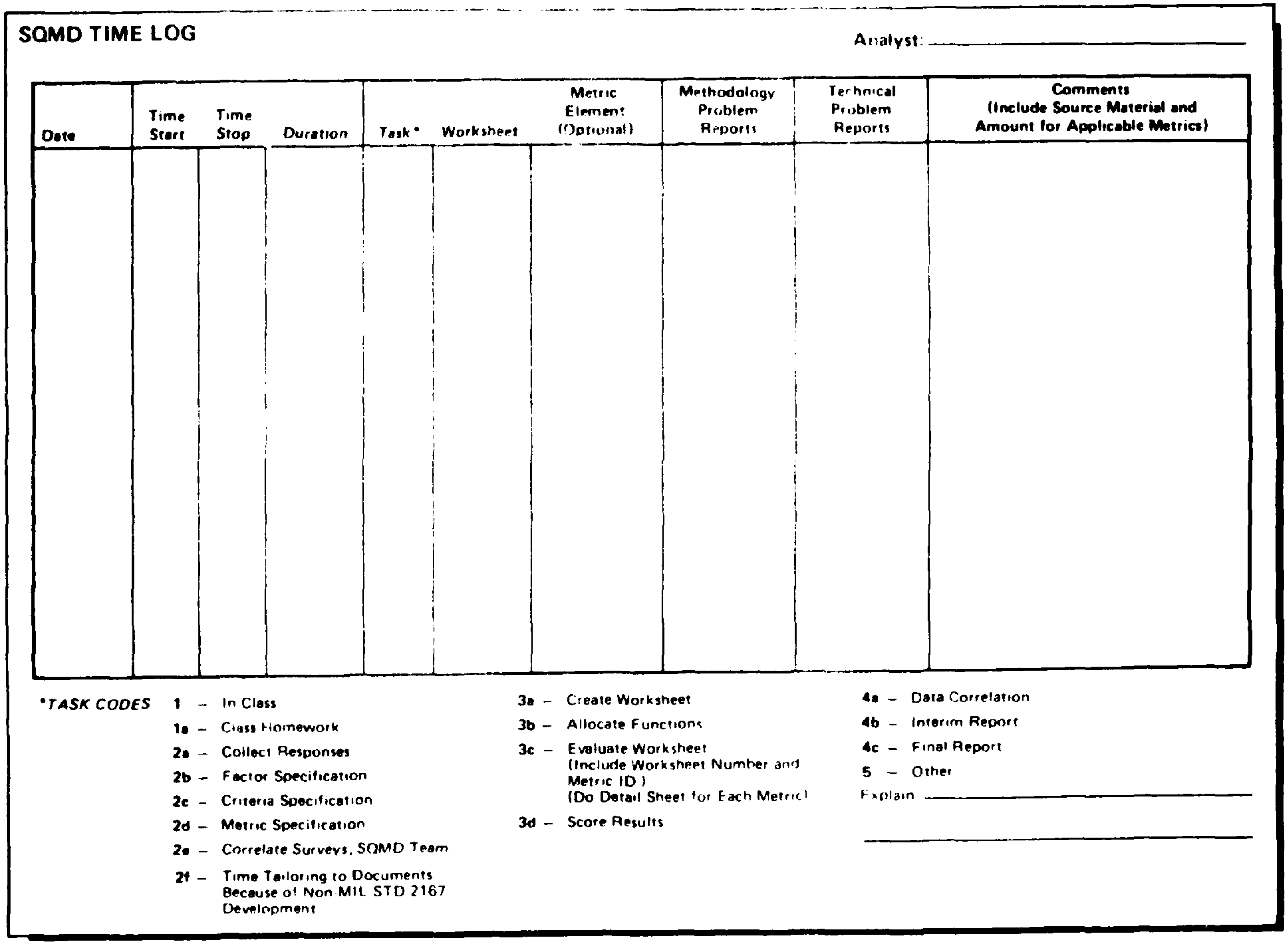
Comments: ____

____

____

____

FIGURE 1.4-3 TECHNICAL PROBLEM REPORT



SQMD TIME LOG

Analyst: ______________________

Date	Time Start	Time Stop	Duration	Task*	Worksheet	Metric Element (Optional)	Methodology Problem Reports	Technical Problem Reports	Comments (Include Source Material and Amount for Applicable Metrics)

*TASK CODES

1 – In Class
1a – Class Homework
2a – Collect Responses
2b – Factor Specification
2c – Criteria Specification
2d – Metric Specification
2e – Correlate Surveys, SQMD Team
2f – Time Tailoring to Documents Because of Non MIL STD 2167 Development

3a – Create Worksheet
3b – Allocate Functions
3c – Evaluate Worksheet (Include Worksheet Number and Metric ID) (Do Detail Sheet for Each Metric)
3d – Score Results

4a – Data Correlation
4b – Interim Report
4c – Final Report
5 – Other

Explain ______________________

____

FIGURE 1.4-4 TIME LOG

selected. Section 3.3 discusses this process in further detail.

Evaluation of compliance to quality requirements used the worksheets shown in Volume III of the guidebook. Quality measurements were taken for each life cycle phase for both test project decision aids. In the ordinary acquisition process, only data selected during quality goal specification would have been collected. Due to the research nature of this effort, however, all metric questions were evaluated and scored. The result was the creation of two sets of quality compliance scores: one using only those factors, criteria, and metrics that were selected and applicable; and one using all metric element questions. Pairs of scores were calculated for each test project decision aid for each software phase. Sections 3.4 and 3.5 contain more details on this process.

Compliance assessment was made based on the quality goals specified for the decision aids. Comparisons were made between achieved scores and project goals. Each metric violation was discussed with project developers to assess its validity and its potential impact on the decision aid systems. Analysis was done on the achieved system quality as compared to the predicted quality assessed at each phase of the life cycle. Section 2.0 presents the results of this assessment.

Based on the data collected and all tasks performed, SAIC analyzed the results to indicate decision aid quality and to evaluate the metric methodology. Sections 2.0 and 3.0 contain the results of this effort, and Section 4.0 presents recommendations and conclusions drawn by SAIC.

1.5 Decision Aid Overview

The decision aids were developed in order to apply decision aid technology to selected tasks of Tactical Air Battle Staff decision making. The purpose of the aids was to aid in planning, designing, demonstrating, and assessing the operational utility and technical feasibility of the applied technology for operational Tactical Air Force personnel. The aids were intended to focus on crisis and wartime decision-making having the potential of materially affecting the outcome of a battle.

To meet these goals, four aids were developed. SAIC has analyzed two of these: the Enemy Sortie Capability Measurement Aid (ESCMA) and the Enemy Course of Action Evaluation Aid (ECOAEA).

The ESCMA was designed to perform quantitative analysis to estimate an enemy's sortie generation capability. It was based on models of how components of installations combine to determine sortie generation, and it identifies combinations of circumstances that can affect this capability. Expected bad weather and serious hangar damage are two example circumstances which might combine to reduce enemy sorties from a particular airfield or for a particular mission.

The ECOAEA allows a user to evaluate various hypotheses on enemy courses of action based on current intelligence gathered about enemy forces, weather, supply needs, etc. It estimates which of the actions would be seen as most favorable from an enemy commander's viewpoint. The aid prompts the user to give subjective inputs and/or importance weights for evaluation factors deemed necessary to perform an analysis on the probability of a certain hypothesis. This information closely resembles that which analysts presently use to form their hypotheses.

After the creation of the aids, they were evaluated by their potential users. Techniques were developed to allow RADC engineers to demonstrate system use and capabilities at RADC.

The main focus of the development of these aids was not to create a system to be fielded, but rather to demonstrate a prototype capability for future deployment in the field. The aids were designed to eventually use existing intelligence data bases, but this initial implementation used sample data and was not intended to actually access the data bases. The development methodology was intended to produce this demonstration prototype system rather than to create a fully maintainable, fielded system.

2.0 QUALITY EVALUATION RESULTS

This section of the report discusses the results of assessing the quality of the Enemy Course of Action Evaluation Aid (ECOAEA) and the Enemy Sortie Capability Measurement Aid (ESCMA). Quality was assessed for each aid at the requirements, design, detailed design, and coding levels. Though measurement was not performed concurrently with the development process, the decision aid systems were the best available test vehicle for the demonstration of the software quality measurement methodology.

In general, measured quality scores did not meet quality specification goals set at the beginning of this project. This is due, we believe, to three separate factors.

The first factor concerns the nature of the development of the decision aid systems. Both aids are proof-of-concept systems that were intended to demonstrate a capability. While the developer wished to produce and deliver a high quality system and specified goals to that level, that achievement was not likely to be within the scope or budget of the effort itself. The main concern was to create a working product, within the time and budget specified by the program office.

The second factor relates to the software quality measurement methodology. The metric evaluation elements were not designed to reflect the quality of decision aid developments. Much of the content of a decision aid or expert-based system lies in the rule base used to drive the conclusions. This rule base did not lend itself to the current metric questions, and was not scored for this assessment. The present state of the art allows algorithms and more standard data structures to be evaluated, but does not adequately address the rule base itself.

The third factor concerns the knowledge of the developers about software quality assessment technology. Software quality goal specification and measurement were not familiar to the engineers involved in the system development and in specifying quality goals. This lack of familiarity resulted in the engineers specifying goals that were above those really required to guarantee that the system be effective and successful.

For both aids, the quality results discussed below include only the scores achieved by evaluating a selected set of applicable metric questions. Because of the research nature of this effort, data was collected for all metric questions. In actual use, however, only those elements applicable to the project and selected at quality goal specification would have been evaluated.

The differences between scores calculated using all metric questions (as was done for research purposes) and scores calculated using only applicable questions indicates the potential differences caused by analyst scoring subjectivity. Figure 2.0-1 illustrates this using the quality factors evaluated for the ECOAEA decision aid on Worksheet 1. The lightly shaded bars on the figure represent scores calculated using only the applicable metric elements, and the solid bars represent scores calculated considering all elements, regardless of applicability.

Applicability is an important attribute to consider. If such choices are left to the individual analyst, wide variations in scoring can result purely from his selection of how to answer some questions. For example, an analyst may decide to answer a particular question as "N/A" rather than as "no" or "0." The metric, criteria, and factor scores will each be higher if "N/A" is selected than if "no" or "0" is used. Because of this subjectivity and potential variations among analysts, SAIC is recommending that procedures be established specifying when the "N/A" answer may be used (see Section 4.4.7).

2.1 Quality Assessment Results

In general, neither decision aid achieved the desired software quality factor goals. Figures 2.1-1 and 2.1-2 present scores for the ECOAEA and the ESCMA, respectively. The factor INTEGRITY shows up as "zero" on the figures because it is not applicable to either decision aid.

The figures do show a scoring trend, however, in that quality seemed to increase during the development cycle. Since the measurement of quality did not take place in parallel with the project development, we cannot attribute this increase in quality to the use of the software quality measurement methodology. We believe that the reason lies partly in the quality and amount of software development documentation available. Both of the aids had few requirements documented, and design was incomplete. This meant that there was relatively little material upon which to calculate quality for the requirements, preliminary design, and detailed design phases. The code was complete, of course, and allowed a [illegible] factor [illegible] calculated which [illegible] a better reflection of [illegible] quality.

One of the purposes of the SQM methodology is to allow assessment of early development quality in order to predict the final system quality. It is interesting to consider further the trends in scores developed by both aids. [illegible] methodology, we would typically [illegible] that low scores in the early phases of development would continue throughout the project. The result would be a system of low measured quality. While a "lowness" trend did continue through the phases for each aid, there was a definite upward trend in the values calculated. It is interesting, therefore, to consider how poorer quality requirements and design documents resulted in higher quality code.

We believe that this was caused by the size of the two decision aids, and by the quality of the people who created them. Smaller systems may be completely understood by the development team, even though the underlying assumptions, goals, and design are not fully documented. The

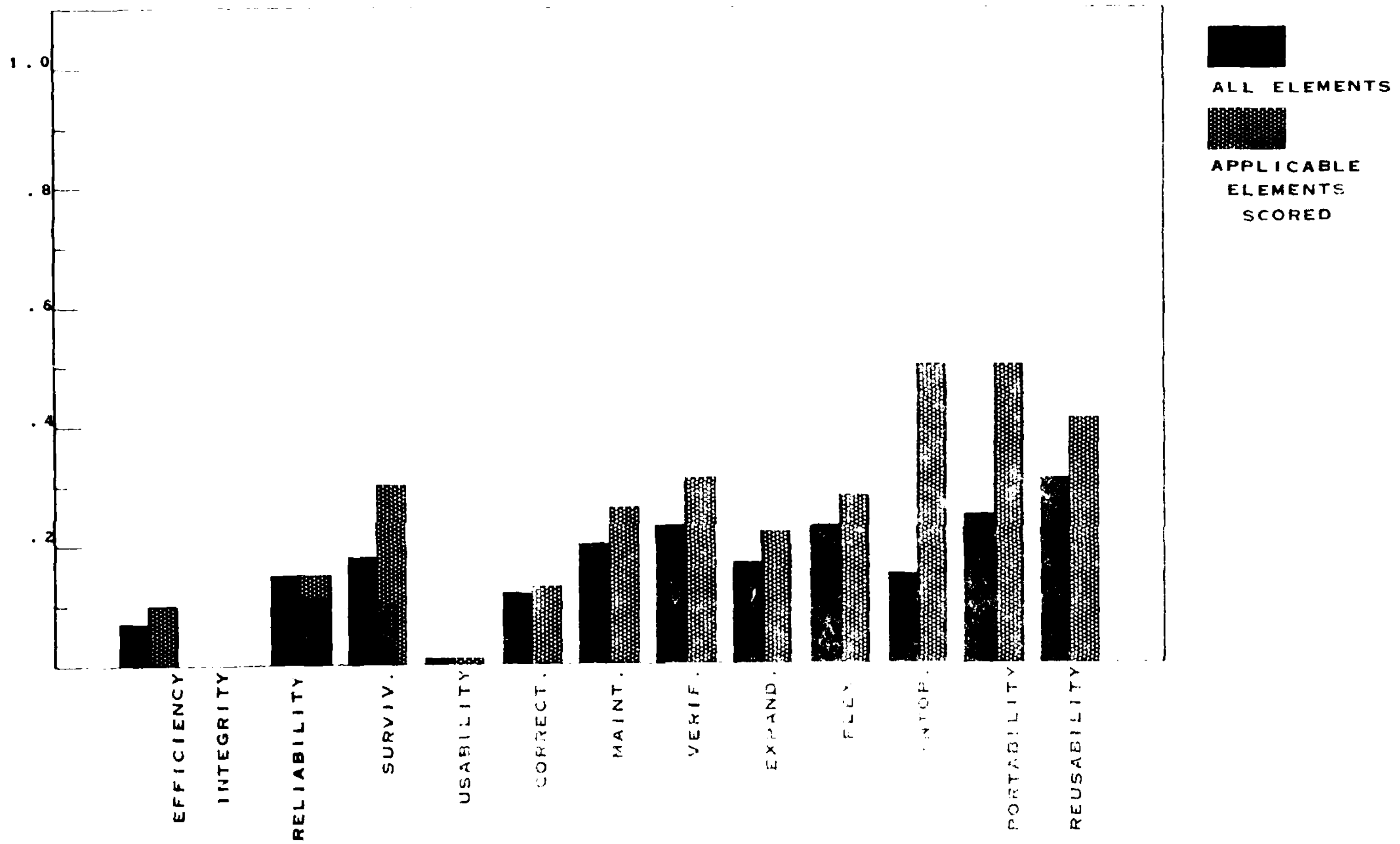


FIGURE 2.0-1 ECOAFA SCORES FOR [illegible]
(REQUIREMENTS PHASE)

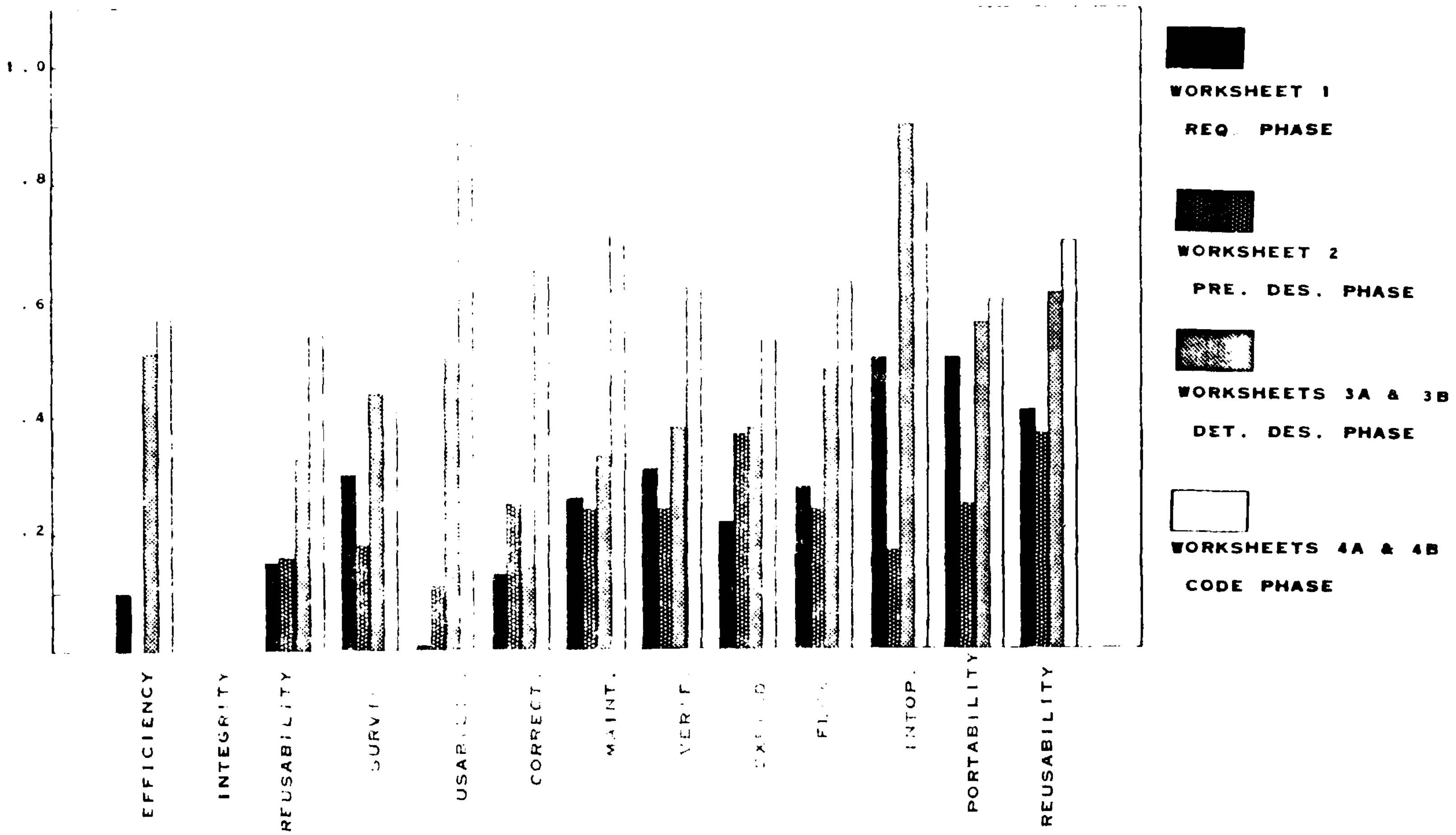


Figure 2.1-1. ECOAEA Factor Scores

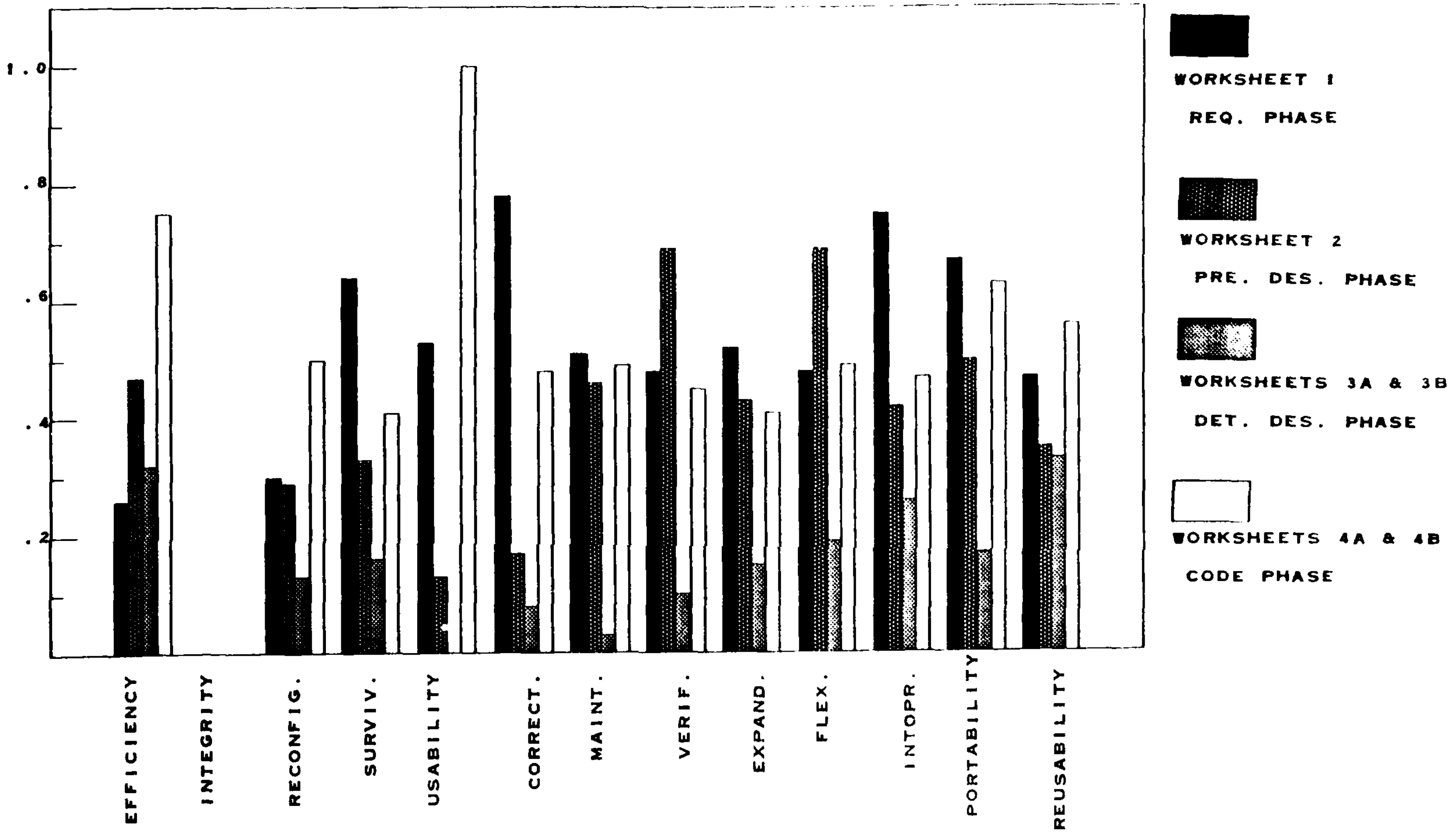


Figure 2.1-2. ESCMA Factor Scores

developers were able to fully understand their tasks, and to perform them even if the documents were lacking. Because of their skill and experience, therefore, the developers were able to create a better quality system than one would expect based only on project documentation. This in no way invalidates the software quality measurement methodology. In a large-scale acquisition, it is less possible for skill and experience to overcome early project deficiencies. It is vital that early quality be high, and that this standard be maintained throughout development of these systems.

2.2 Technical Problems Uncovered

On the requirements and preliminary design worksheets, SAIC created technical problem reports whenever a metric violation was uncovered. Time and budget did not permit the generation of these reports for violations uncovered during detailed design and coding worksheet evaluation.

The reason for generating technical problem reports was to gather information concerning the developer's responses to the problems that were discovered. In all, 249 reports were written. Of these, 105 concerned questions that were actually not applicable to the decision aid systems or concerned non-existent standards, and no developer response was requested.

The developer responded to the remaining 144 comments, as shown below:

- 1 technical problem was identified as being both valid and of a critical nature. This problem concerned the general quality of the documentation, and particularly the discrepancy between the designed units and the coded units. The developer agreed with the problem report, and identified the correction as of critical impact to the system. Maintainability, in particular, was described as being lowered by the document quality.

- 5 technical problems were identified as valid and as having a moderate impact on the quality of the system.

- 109 problems were identified as valid, but with only slight impact on the quality of the two decision aids.

- 24 problems were identified as valid, but of no impact on the quality of either aid. This was due chiefly to the size of the two decision aid systems.

- 5 technical problems were identified as invalid, and as having no meaning with respect to the quality of the decision aids.

SAIC agrees with the evaluation responses made by the developer.

A sample of the developers comments on the problems uncovered are included below. Both comments were categorized as valid, and as having moderate impact on the quality of the system.

Concerning Worksheet 2, Metrics AT.1(3), AT.2(3), and AT.3(1):
[Questions concern auxiliary storage space allocated, processing time allocated, and I/O channel time allocated]

Providing more detailed documentation on the allocations [of time and space] at the outset might be counterproductive because, as the [SAIC] analyst suggests, of the prototype nature of the aid -- and because of the comparatively rich facilities (in particular, the VAX 11/750) available. It could be demonstrated at the outset that time and space allocations will not be a problem: even a distinctly suboptimal preliminary design would be adequate to prove that point. ... The real interest lies in making good-- not just adequate — time and space allocations, in particular with a view to subsequent porting to a microcomputer. Concurrent system design and prototyping, with each activity informing the other, is an excellent way of achieving good time and space allocations when, as in the present case, neither time nor space requirements are overly constrained, and the overall program is a relatively small one. ... Knowing that one is going to have to formalize such observations at some point leads to a raising of the consciousness that may indeed prove beneficial.

Concerning Worksheet 2, AM.3(1): [Recovery from computational failures]

Many failures and errors can be covered by standard programming techniques and should not have to be addressed individually at the program design level.

2.3 Score Validation

The decision developers conducted testing to verify the validity of some of the quality factor scores. In particular, they looked at REUSABILITY, PORTABILITY, and the criterion anomaly management.

2.3.1 ESCMA

Limited time was available to the developers to review the 15,000 lines of ESCMA code. Since the decision aid was actually developed by a sub-contractor (Betac), PAR had no access to object code nor to a compiler. The comments below should be understood in the light that they are the best analysis possible, but it was impossible to check the results presented in many ways that would have been possible if the object code or a compiler had been available.

Anomaly Management. All file access within ESCMA is handled by instructions that are built into Pascal. No special provisions for file-access error handling were included in the code. Accordingly, all such errors are dealt in default ways determined by the interaction of the compiled code and the operating system.

No checks were found for data errors from files. While no underflow or overflow checking was done within the Pascal code, the linear programming pacakge used with the system does make such checks.

REUSABILITY. Several of the sections of code could be reused in other applications. This includes the linear programming package, keyboard procedures, and screen procedures. This includes approximately 15% of the code. No other reusable code was detected.

2.3.2 ECOAEA

The ECOAEA was written in-house by the developers, and for these tests they had full accessibility to the code and its modifications.

Anomaly Management. There are several routines in the aid that handle missing data files. If a file is non-existent, an error routine is called. The routine prints a message, and then terminates the process. ECOAEA does not check for the validity of the values it reads or uses. For this particular aid, most of the values are originally primitive input values which are then aggregated. As a prototype, ECOAEA assumed the input data values to be correct, with the assumption that any produced values were then also valid. All user inputs, other than the <BREAK> key, are trapped and handled by the program, but there is no checking for overflow or underflow.

REUSABILITY. There is a routine (get_option in file windows.c) which returns user inputs at menu selections. It is easily reusable. This piece of code has been used by the developer in a variety of projects, with only minor modifications. In addition, the entire file (colors.c) is a generic version of the UNIX cursor package. Any software which uses a VT-220 or compatible terminal will be able to use the tools which this file provides. These tools include code for moving the cursor to a specific position, clearing a line, clearing a window, for inverse video, and for specifying foreground and background colors.

PORTABILITY. The basic underlying software of ECOAEA is currently being used in another application by the developer. This application involved porting code from its original development area (VAX 11/750 under UNIX, 32-bit machine) to an Intel 310 machine, running XENIX in a 16-bit environment. The following is a brief description of the amount of time and effort spent in transporting.

The single biggest problem the developer had to deal with was the memory limitation and model of the Intel machine. PAR spent 3-4 days in recompiling the pieces of the source code with appropriate memory models. The main problem encountered here was redefining the sizes of the various data structures used. The Intel machine is byte-addressable, meaning some fields (i.e., integers and floating point) needed to be aligned on even number addresses. Additionally, problems were encountered when a structure size was an odd number of bytes. Many of these problems were not apparent at compilation time, and only careful debugging work allowed them to be solved.

The other serious problem the developer encountered concerned integer length. "C" provides facilities for declaring integers in three different ways. These are INT X; SHORT INT X; and LONG INT X;. By definition, the last two of these declarations will result in integers of equal length. For the VAX environment, the first and third declarations result in integers of 4 bytes, while the second in an integer of 2 bytes. The Intel machine set aside 2 bytes for the first and second forms, and 4 bytes for the third. A good deal of time was spent changing many instances of the first form to the third. Once again, these problems were not discovered until actual execution of the program revealed some suspicious results.

2.3.3 Conclusions

Based on input from the developer, SAIC has concluded that the measured quality results do roughly coriespond to those as supplied by the developer (and as subjectively assessed by the evaluation teams). However, these results are still in the realm of excellent, good, average, and below. Research needs to continue to support establishing the relationship between measured quality and real-world results.

3.0 METHODOLOGY EVALUATION RESULTS

This section of the technical report describes in detail the results of analysis and project efforts concerning the software quality measurement (SQM) methodology itself. The section is organized to reflect the metric application process, as follows:

- Section 3.1 contains results that apply to the entire quality specification and evaluation process.
- Section 3.2 describes in detail the analysis of the class SAIC conducted to train personnel not yet experienced with software metrics. It also discusses differences found between the analysis efforts of the experienced and the inexperienced teams.
- Section 3.3 discusses the process of the specification of software quality goals by the acquisition manager.
- Section 3.4 contains the results of the application of metric element worksheet questions to the decision aid documentation.
- Section 3.5 describes metric scoring as performed on the decision aids.

Throughout discussion of these results, references are made to the recommendations and conclusions contained in Section 4. For each problem uncovered, there is a recommended solution in Section 4.

3.1 General Results

This section presents some very generalized results of the methodology evaluation process. It first discusses some overall observations, and then presents information on the labor effort required to perform the quality evaluation on the two decision aid systems.

3.1.1 General Observations

SAIC followed each of the steps of the software quality measurement methodology presented in the guidebooks. In general, we found the methodology to be sound. The procedures and steps to be performed are basically logical and meaningful, and yield meaningful results. On the more negative side, we found that details were often presented in a confusing fashion. The mixture of justification, theory, procedures, and examples did not do full justice to any part of the methodology. The metric evaluation elements or questions were, in particular, found to be ambiguous and confusing.

3.1.2 Methodology Labor Effort

As each step of the software quality measurement methodology was performed, labor records were kept to indicate how long each took. A total of 852 hours were spent in specifying quality goals and evaluating quality for the two decision aids. Figure 3.1-1 shows how the effort was distributed among the major tasks contained in the guidebook methodology for quality goal specification and quality evaluation.

The labor required to perform the decision aid quality assessment is more meaningful when considered against the amount of material analyzed. Table 3.1-1 shows the size of each of the products analyzed during the quality assessment procedures.

As was expected, the most time-consuming aspect of this process is the evaluation of worksheets and collection of metric data. Early worksheets (0, 1, and 2) were applied only once, and took relatively little time to complete. The final worksheets (3A & 3B, and 4A & 4B), applied to each unit in the system, were quite time-consuming. The 3A & 3B worksheets were faster to evaluate than were the 4A & 4B worksheets, chiefly because not every coded unit was included in the design of either of the two aids.

We have also correlated the labor effort required to evaluate the worksheets based on criteria and factors. Tables 3.1-2 and 3.1-3 present the amount of time required to collect quality factor data for each worksheet, and the total across all worksheets for each aid. Figures 3.1-2 through 3.1-4 are graphical representations of the time required to evaluate factors for both aids together (Figure 3.1-2), for [illegible]A alone (Figure 3.1-3), and for ECOAEA alone (Figure 3.1-4). Appendix B presents the information for the software quality criteria. This may be more meaningful because criteria which are applicable to more than one factor are counted more than once in collecting the factor evaluation time data.

We believe that the differences in time required between the two decision aids is chiefly due to the size of the documents involved and the amount final source code produced. The same tailoring process was used on each aid, so that there were no differences in the number of metric questions answered. While each aid was produced by a different contractor, they are of similar natures and similar measured quality, so we believe no major differences were introduced in that regard.

In addition to evaluating factor and criterion times, we also calculated how long each software unit took to evaluate during application of Worksheet 4B. Figure 3.1-5 represents the time it took to evaluate each unit in the ECOAEA as plotted against unit size. As expected, there seems to be a strong relationship between the size of the unit and how long it takes to answer questions concerning that unit. The points **below the trend line represents units typically more complex than other units. Conversely, those above represent less complex units.**

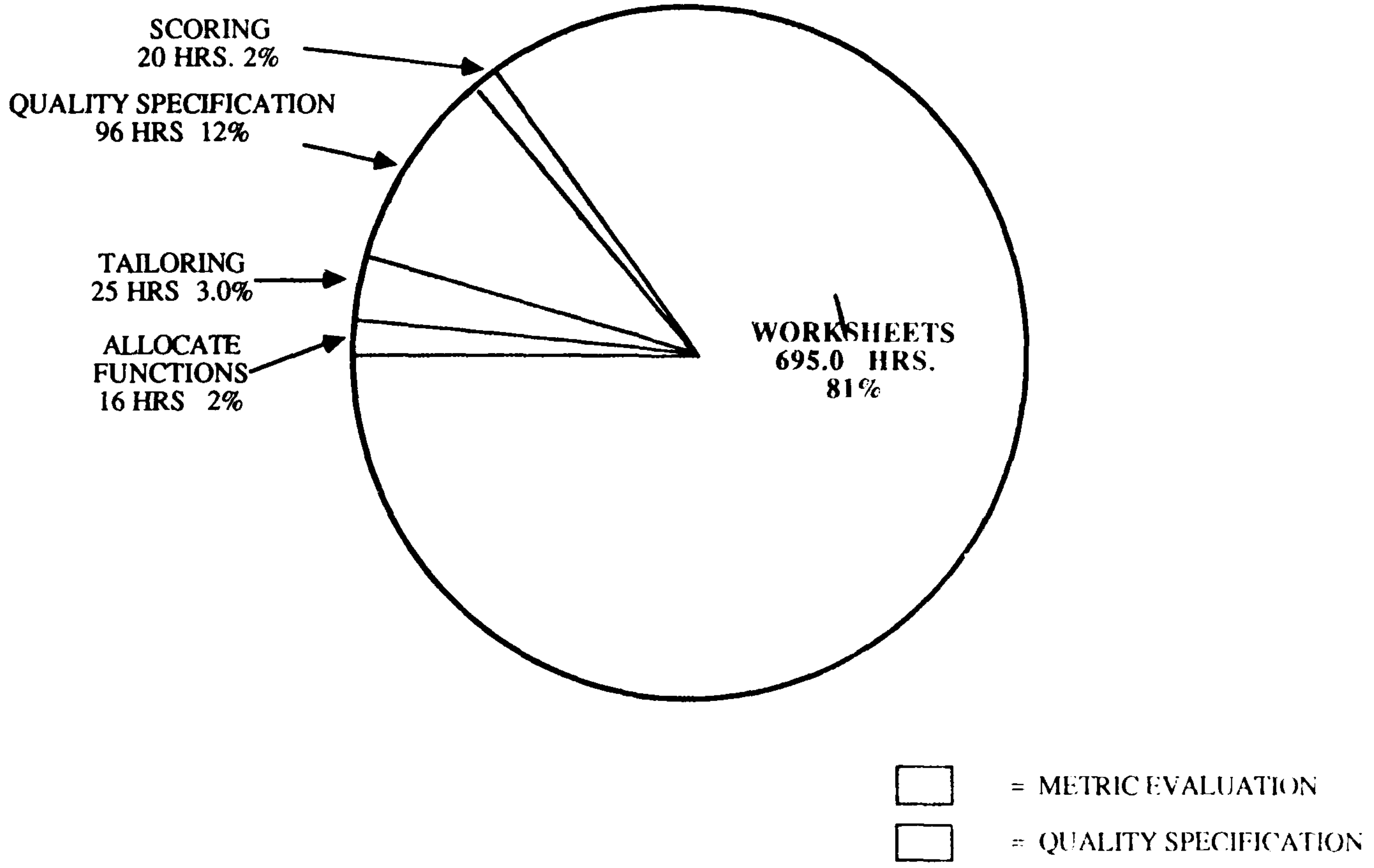


FIGURE 3.1-1 LABOR COMPOSITION BY TASK

TABLE 3.1-1 DECISION AID DOCUMENT SIZE

WORKSHEET AND PRODUCT	DECISION AID	
	ENEMY COURSE OF ACTION EVALUATION AID	ENEMY SORTIE CAPABILITY MEASUREMENT AID
WORKSHEET 1	66 PAGES (FUNCTIONAL DESCRIPTION)	142 PAGES (FUNCTIONAL DESCRIPTION)
WORKSHEET 2	66 PAGES (FUNCTIONAL DESCRIPTION)	142 PAGES (FUNCTIONAL DESCRIPTION)
WORKSHEET 3	55 PAGES OF PDL AND DATA DIAGRAMS	90 PAGES OF PDL AND DATA DICTIONARY
WORKSHEET 4	3,334 LINES OF CODE	13,575 LINES OF CODE

TABLE 3.1-2 TIME TO COLLECT QUALITY FACTOR DATA FOR ESCMA (BY WORKSHEET IN MINUTES)

QUALITY FACTOR	WORKSHEET				TOTAL
	1	2	3 (A+B)	4 (A+B)	
EFFICIENCY	143	45	520	944.3	1652.3
INTEGRITY	70	18	N/A	N/A	88.0
RELIABILITY	751	203	73.0	3196.3	4223.0
SURVIVABILITY	681	367	95.4	856.8	2000.2
USABILITY	513	131	11.0	67.0	722.0
CORRECTNESS	717	151	92.5	503.5	1464.0
MAINTAINABILITY	385	123	116.4	4653.8	5278.2
VERIFIABILITY	219	46	83.1	4580.9	4929.0
EXPANDABILITY	361	130	126.1	4973.9	5591.0
FLEXIBILITY	219	46	94.5	4828.3	5187.8
INTEROPERABILITY	905	269	62.0	1320.9	2556.9
PORTABILITY	163	41	58.4	2188.4	2450.8
REUSABILITY	715	348	323.8	7831.2	9218.0

TABLE 3.1-3 TIME TO COLLECT QUALITY FACTOR DATA FOR ECOAEA (BY WORKSHEET IN MINUTES)

QUALITY FACTOR	WORKSHEET				TOTAL
	1	2	3 (A+B)	4 (A+B)	
EFFICIENCY	165	95	242.8	335.5	838.3
INTEGRITY	5	2	N/A	N/A	7.0
RELIABILITY	244	170	518.3	1107.1	2039.0
SURVIVABILITY	526	272	281.1	559.5	1638.6
USABILITY	275	83	10.8	60.8	429.6
CORRFCTNESS	198	123	243.1	211.0	775.1
MAINTAINABILITY	235	137	933.0	1703.8	3008.8
VERIFIABILITY	135	82	863.5	1663.2	2743.7
EXPANDABILITY	198	125	822.3	1939.5	3084.8
FLIXIBILITY	135	82	814.5	1835.7	2867.2
INTEROPERABILITY	330	89	294.1	673.5	1386.6
PORTABILITY	160	69	299.0	832.8	1360.8
REUSABILITY	699	274	1205.3	2875.6	5053.9

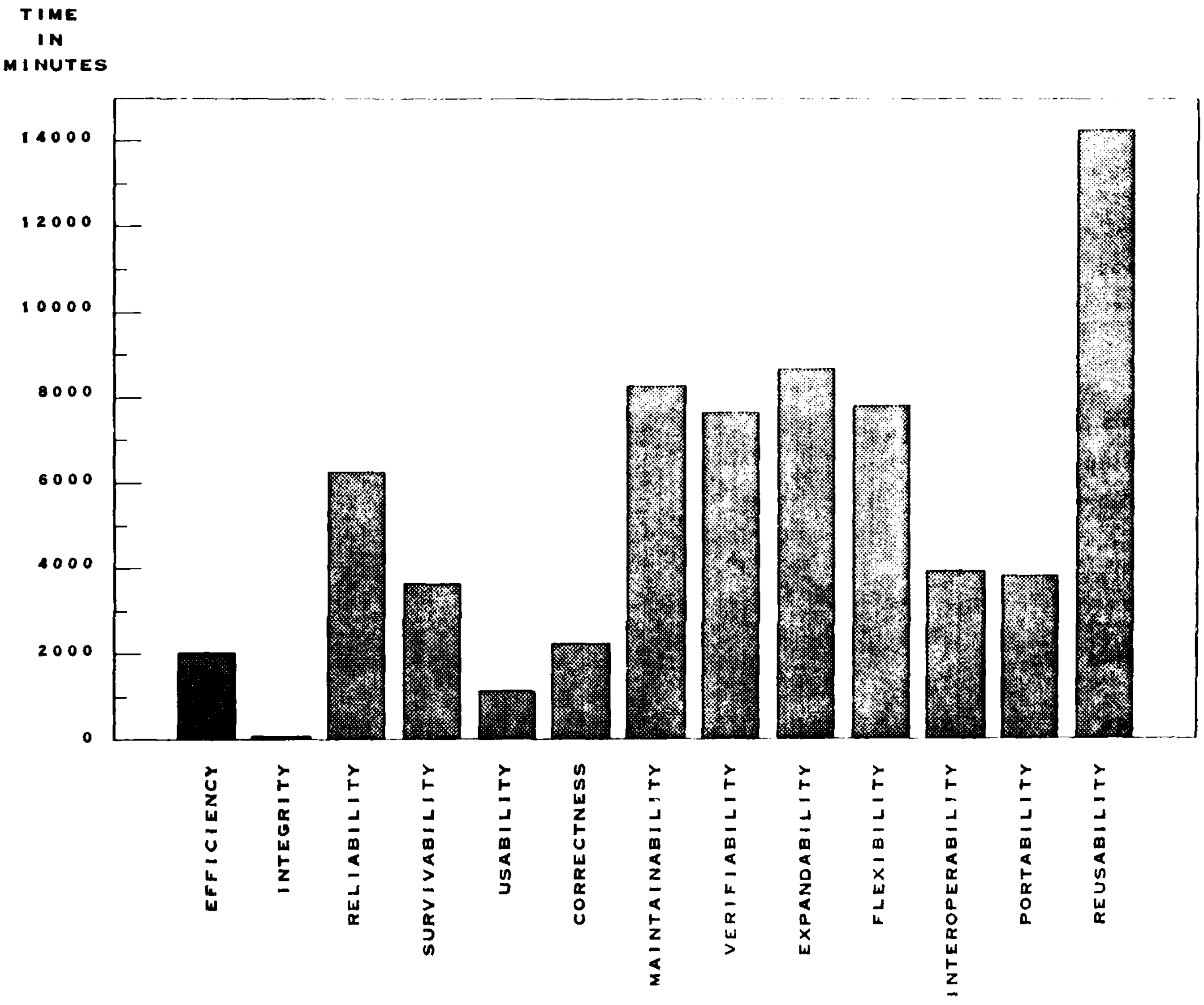


Figure 3.1-2. Factor Evaluation Time for Both Decision Aids

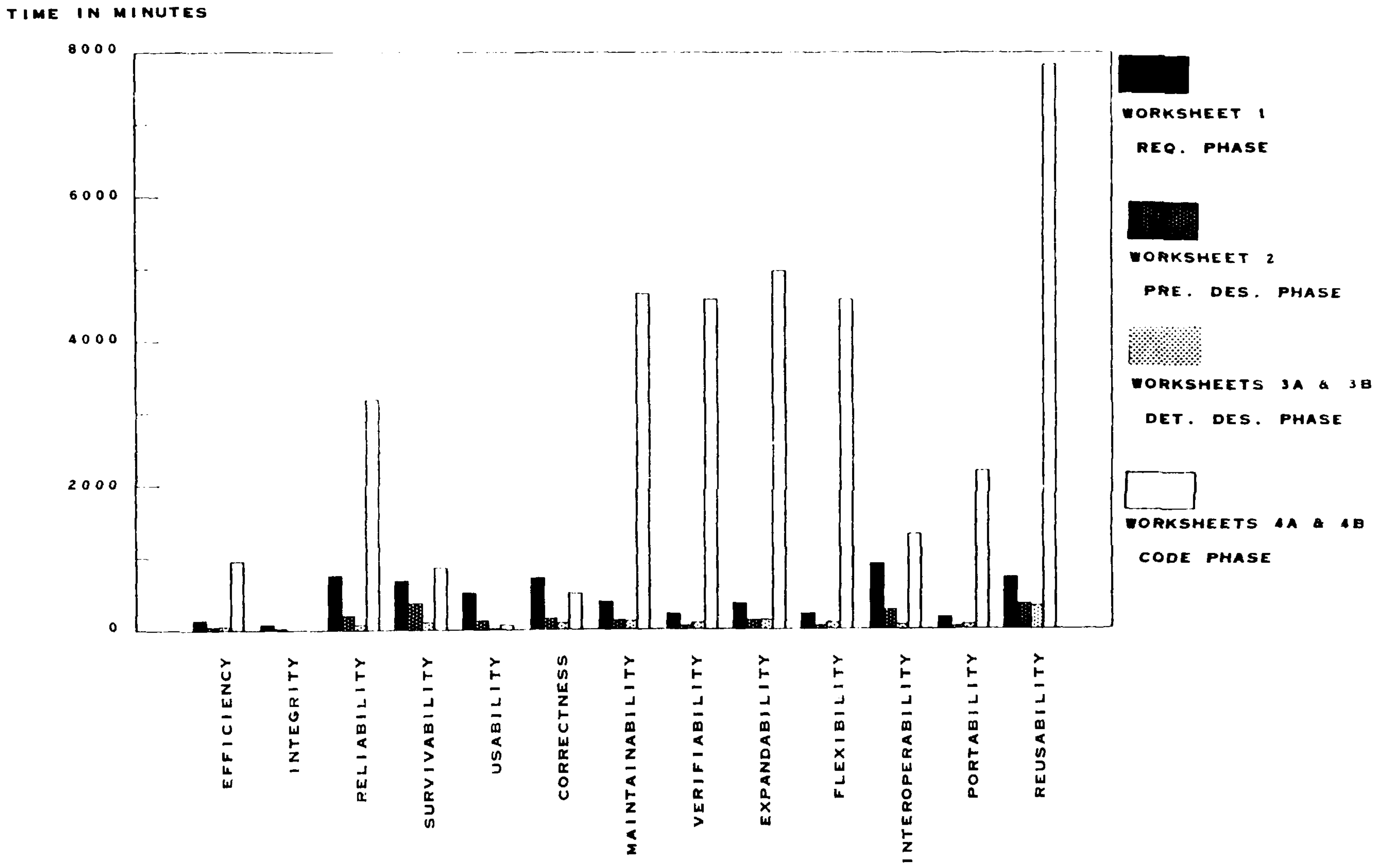


Figure 3.1-3. Factor Evaluation Time for ESCMA

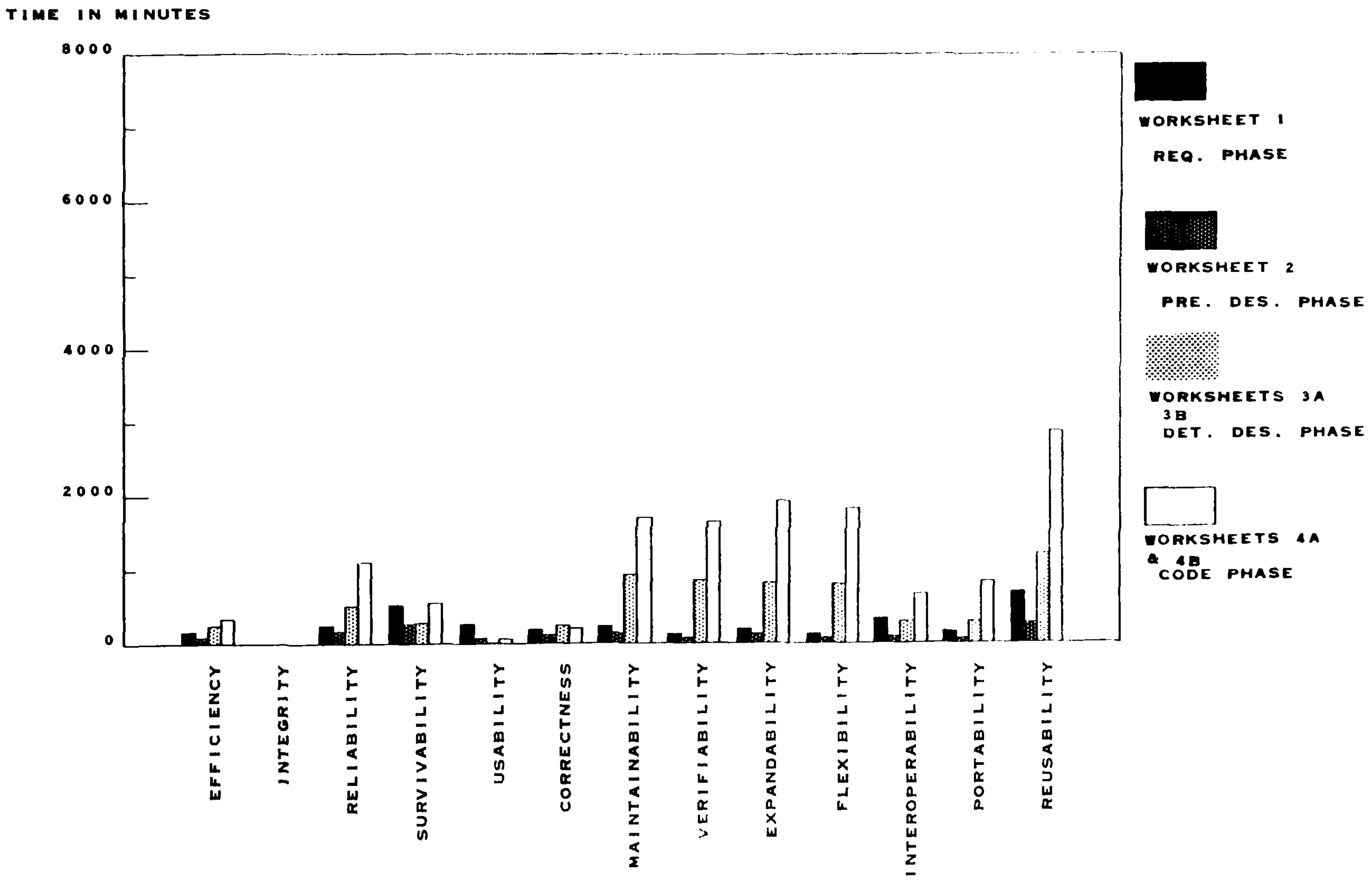


Figure 3.1-4. Factor Evaluation Time for ECOAEA

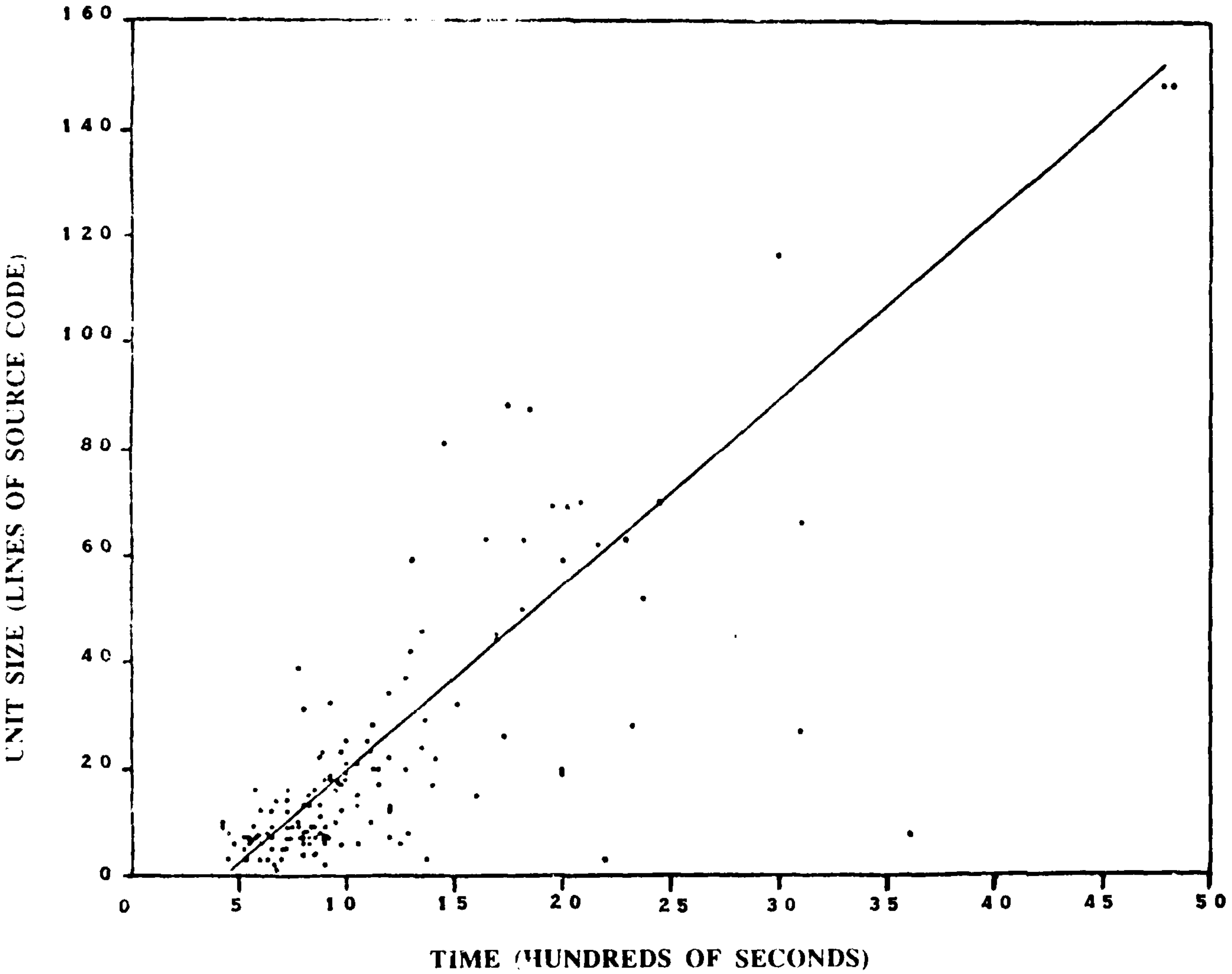


FIGURE 3.1-5 PLOT OF MODULE EVALUATION TIME AGAINST SIZE

3.2 Experienced/Inexperienced Project Members

As part of the project, SAIC utilized two separate teams of analysts. Each team evaluated one of the two decision aids used as test projects. The teams varied in experience, with one team new to metric application and one team consisting of personnel who had worked with metric elements on other projects.

3.2.1 Class Evaluation

Part of SAIC's approach to the project involved the training of the inexperienced project members. This training consisted of a formal class presented over a three-day period. The class was intended to familiarize students with the metric framework, and with the SQM methodology specifically. The class consisted of lecture/question sessions, and included outside "homework" in the form of workbooks. Figure 3.2-1 presents the outline of the class as taught.

Though the class had originally been planned only for those new to software measurement technology, each of the more experienced team members expressed interest in attending. Each experienced person had worked on other projects involving software metrics, but felt that his knowledge was incomplete. After discussion of project goals, it was decided that all members should attend the class. The presence of experienced personnel gave the "newcomers" the benefit of real-world experiences during class discussions. In addition, the common information and focus provided a single unified starting point for all project tasks. It was also noted that even though experienced, the "oldtimers" did not know everything discussed in the software quality measurement methodology guidebooks. The training provided them with new information to be used on the project.

After class completion, while each team member was actively working on the specification of quality goals or metric application, each attendee was asked to evaluate the effectiveness and quality of the class sessions. Figure 3.2-2 is the form each student was asked to use. Forms were not filled out by class attendees who subsequently left the project. Section 3.2.3 discusses project personnel further.

Table 3.2-1 presents the ratings given to the class by the students. These ratings are on a scale of 1 to 5, with 5 meaning most effective. The ratings do not correlate directly with levels of experience or areas of expertise. Comments did consistently show, however, that the students desired examples and practice with actual metric question evaluation as part of the class.

After training was completed, two junior analysts joined the project. Both were inexperienced with software measurement methodologies and were placed on the "inexperienced" team. Training of these analysts was conducted informally by other project personnel. The result of the informal training was that while able to complete all tasks, the two

DAY 1:

- Introduction to Class
  - Class Outline
  - List of Records to Maintain
- Introduction to Metric Technology
  - Quality Specification
  - Quality Evaluation
  - System Life Cycle
- Life Cycle Approach
  - Quality Factor Requirements
  - Quality Level of Detail
  - Goal Levels
  - Criterion Attributes
  - Evaluation Report
- Mea[illegible]
  - Hierarchical Quality Model
  - User Concerns for Quality Factors
  - Factor Relationships
  - [illegible] of Terminology

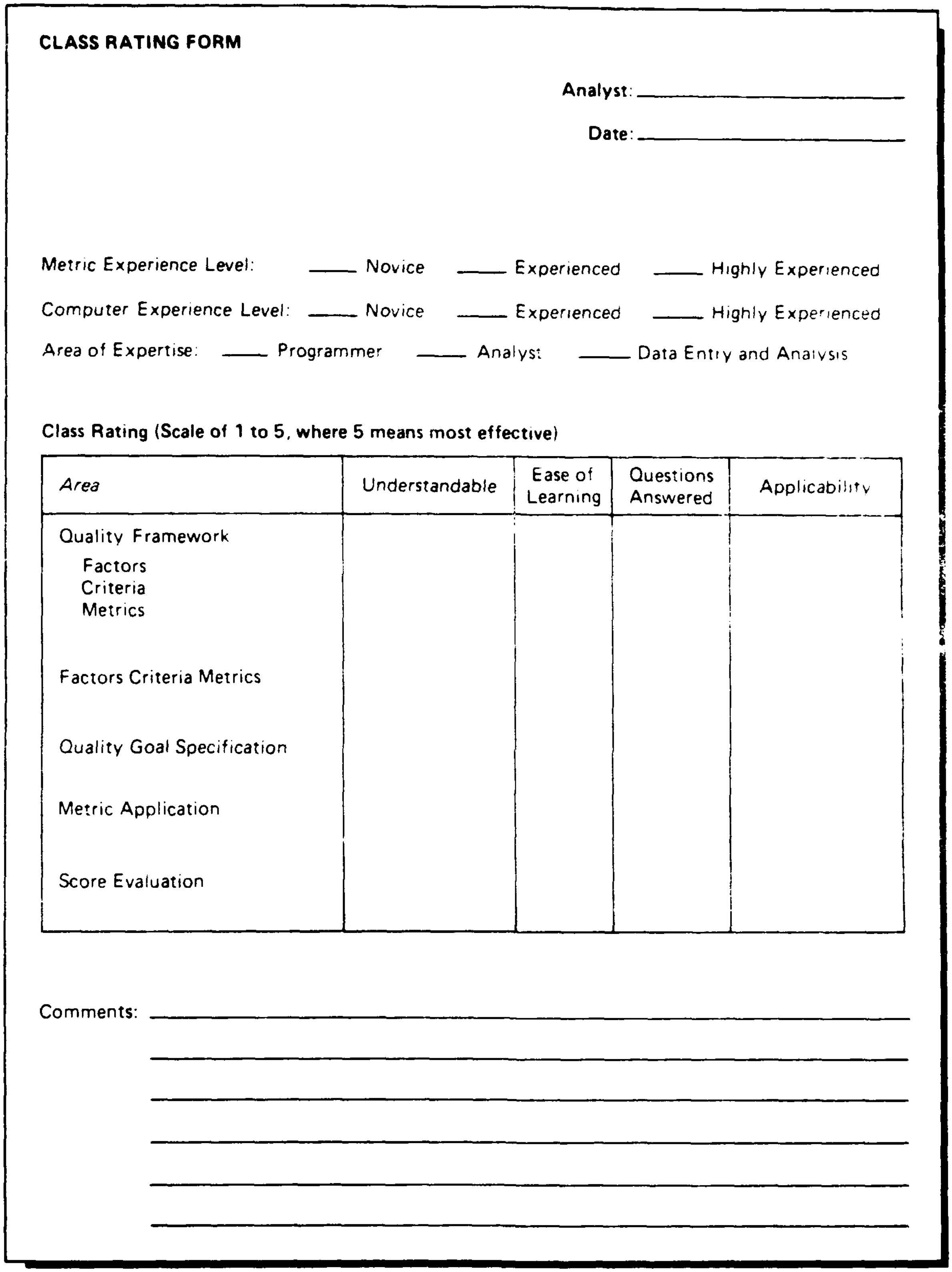
DAY 2:

- Quality Specification
  - Select and Specify Factors
    - System Function
    - Quality Goals
    - Interrelationships
    - Costs
    - Review Goals – Modify
  - Select and Specify Criteria
    - Criteria Goals
    - Interrelationships
    - Weighting
    - Review
  - Select and Specify Metrics
    - Metrics
    - Metric Elements
- Quality Evaluation
  - Identify Allocation Relationships
    - System, CSCI, Unit
    - Allocation to Software Elements
    - Evaluation Formula
  - Select Phase Level
    - Prepare Worksheet
    - List Source Material
  - Apply Worksheets
    - Application Guide
    - M[illegible]S
    - Violations and Deficiencies
- Score and Analyze
  - Score Factors
  - Analyze Scoring
  - Recommend Corrective Action

DAY 3:

- Working Session
  - Analyze Workbook Questions
  - Sample Applications

FIGURE 3.2-1 TRAINING CLASS OUTLINE



CLASS RATING FORM

Analyst: ____________________

Date: ____________________

Metric Experience Level: _____ Novice _____ Experienced _____ Highly Experienced

Computer Experience Level: _____ Novice _____ Experienced _____ Highly Experienced

Area of Expertise: _____ Programmer _____ Analyst _____ Data Entry and Analysis

Class Rating (Scale of 1 to 5, where 5 means most effective)

Area	Understandable	Ease of Learning	Questions Answered	Applicability
Quality Framework Factors Criteria Metrics				
Factors Criteria Metrics				
Quality Goal Specification				
Metric Application				
Score Evaluation				

Comments: ____

____

____

____

____

____

FIGURE 3.2-2 CLASS EVALUATION FORM

TABLE 3.2-1 CLASS EVALUATION RESULTS

CLASS ELEMENT	STUDENT 1	STUDENT 2	STUDENT 3	STUDENT 4
Metric Experience	Experienced	Novice	Experienced	Novice
Computer Experience	Experienced	Experienced	Novice	Experienced
Area	Programmer	Programmer	Analyst	Programmer
CLASS RATINGS:				
Framework	5.0	4.5	4.5	5.0
Factors Criteria Metrics	5.0	4.5	2.0	5.0
Goal Specification	3.5	3.5	4.0	3.0
Metric Application	3.7	4.0	3.7	3.0
[illegible]	3.7	4.0	4.7	3.0

junior analysts had a much less complete idea of the process of metric specification and evaluation. They had more difficulty understanding the reasons behind some of the steps. They also tended to believe that their own lack of understanding was causing a problem, even if the problem was actually in the methodology itself. This was particularly evident in worksheet evaluation. Neither analyst produced many Methodology Problem Reports when they encountered difficulties with a metric element question. Instead, each believed that any problem was in his own lack of understanding, not in the question itself. Since other analysts examined and evaluated every metric question as well, this did not impact SAIC analysis efforts.

Based on SAIC's experience with training and with metric applications, we are recommending that training (either in the form of classes or of prepared materials) be made available to both acquisition managers and to those who will be evaluating metric compliance. This is discussed in paragraph 4.2.3.

3.2.2 Differences Caused by Experience

At the beginning of the project, we expected certain differences to show up between the more experienced and less experienced project analysts. We anticipated that the less experienced personnel would perform quality specification and assessment tasks less quickly than would the more experienced. The less experienced team was expected to have more questions, and to understand the evaluation process less completely.

These expected differences proved to be correct. Table 3.2-2 contains some of the differences between the experienced and inexperienced project personnel in performing software quality evaluation tasks. The more experienced project personnel were faster at metric question evaluation and generated fewer problem reports concerning the methodology itself.

The less experienced team did, in addition, reach frustration levels earlier than did the more experienced team. When questions were particularly confusing or ambiguous, or the documentation being analyzed particularly deficient, the less experienced analysts found the experience frustrating. The other team was better able to deal with these problems.

3.2.3 Project Personnel

Figure 3.2-3 presents both the original organization of the project team and the revised organization used throughout most of the project. The team evaluating the Enemy Sortie Capability Measurement Aid, which is written in Pascal, was selected because of their experience with that language. The team evaluating the Enemy Course of Action Evaluation Aid was selected because of familiarity with the language it is written in, "C."

TABLE 3.2-2 DIFFERENCES FOR EXPERIENCED AND INEXPERIENCED PERSONNEL

AREA	EXPERIENCED PERSONNEL	INEXPERIENCED PERSONNEL
[illegible] OF ANSWERING QUESTIONS [illegible] ON [illegible] SIZE	[illegible] MINUTES/LINE OF CODE	[illegible] MINUTES/LINE OF CODE
[illegible]	[illegible] LINES OF CODE	[illegible]
[illegible]	[illegible] PAGES	[illegible] PAGES
[illegible]		

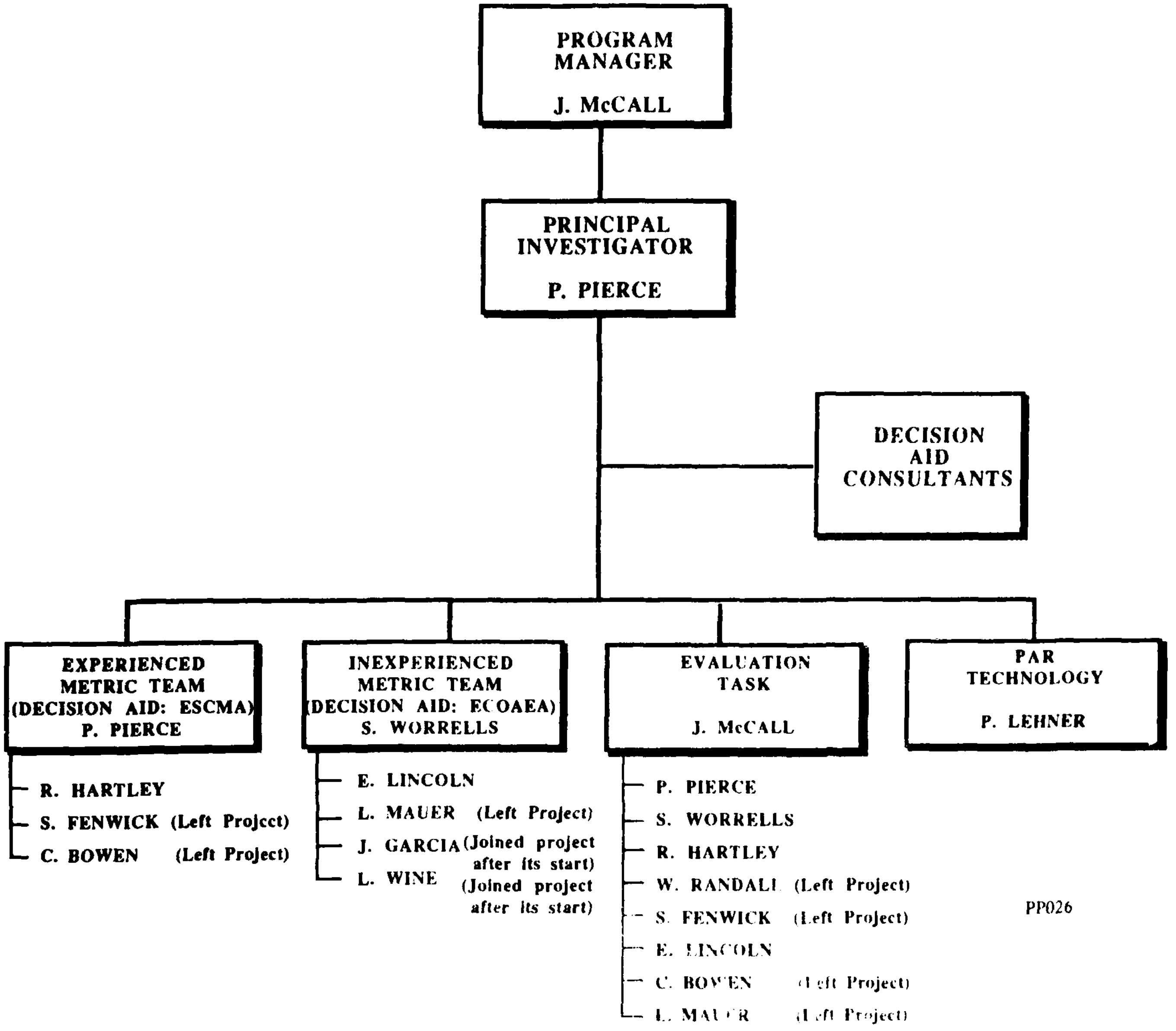


FIGURE 3.2-3 PROJECT ORGANIZATION

Some personnel changes took place over the life of the project. Ms. Cindy Bowen, Ms. Susan Fenwick, Ms. Linda Mauer, and Mr. Bill Randall left the project before worksheet evaluation began. Two junior analysts, Mr. John Garcia and Ms. Louise Wine, joined the project in their place and were assigned to the inexperienced project team.

3.3 Software Quality Requirements Specification

The first step in the software quality measurement (SQM) methodology involves the specification of quality requirements by the acquisition manager. These requirements include those applicable to the quality factors, the criteria to be used and their relative weightings, and the selection of the applicable metric elements to be evaluated on the various worksheets.

The methodology, results, and problems encountered during the application of quality goal specification are described in this section. The material is organized as follows:

- Section 3.3.1 describes the steps followed in performing the goal specification and metric selection tasks.
- Section 3.3.2 describes the results and problems uncovered in specifying factor quality goals for the decision aid systems.
- Section 3.3.3 describes the process of selecting and weighting the quality criteria. It includes discussion of the results for each aid, and the problems uncovered.
- Section 3.3.4 discusses the selection of the metric elements applicable to the decision aids for all worksheets. Results and problems are included.
- Section 3.3.5 analyzes the cost of quality specification

As with other parts of Section 3.0, no specific recommendations or conclusions are drawn from these results. These are included instead in Section [illegible], and are cross-referenced to the problems uncovered within these paragraphs.

3.3.1 Methodology Description

The software quality measurement (SQM) [illegible] the following steps in the process of specifying quality requirements:

- Select and Specify Quality Factors
  - Identify functions
  - Assign quality factors and goals
  - Consider interrelationships
  - Consider costs

- Select and Specify Quality Criteria
  - Select criteria
  - Assign weighting formulas
  - Consider interrelationships

- Select and Specify Quality Metrics
  - Identify metrics
  - Select and qualify metric elements

The result of this process is the specification of the quality goals a system is to meet. The process is described in detail in Volume II of the guidebooks [BOE-2], in paragraphs 4.1, 4.2, and 4.3. Paragraphs below describe the way SAIC performed each of these steps in compliance with the SQM methodology, together with the results obtained.

3.3.2 Select and Specify Quality Factors

To specify quality factors, SAIC used the quality goal survey questionnaire described in the software quality measurement methodology guidebooks. This survey is designed to afford a means of determining desirable quality goals as seen by the potential users and acquisition managers. For the decision aid project, surveys were sent to the original system developer (PAR) and to the SAIC team members working on this effort. In addition, each was asked to fill out a response form indicating their reactions to the validity and conduct of the survey. The survey questionnaire is presented in Appendix A.

Table 3.3-1 indicates the survey results for each respondee for each decision aid. The quality goals to be specified by each user were of the form Excellent, Good, Average, and Not Applicable. These goals correspond to numeric quality factor scores of 0 (Not Applicable), .7 to .8 (Average), .8 to .9 (Good), and .9 to 1.0 (Excellent).

TABLE 3.3-1 QUALITY GOAL SURVEY RESULTS

QUALITY FACTOR	ESCMA			ECOAEA			
	Developer	Pierce	Hartley	Developer	Lincoln	Wine	Worrells
EFFICIENCY	A	G	G	E	A	N/A	A
INTEGRITY	N/A	N/A	N/A	N/A	A	G	E
RELIABILITY	G	E	E	E	E	E	E
SURVIVABILITY	N/A	A	N/A	N/A	G	N/A	A
USABILITY	E	E	E	E	G	E	G
CORRECTNESS	G	E	E	E	E	E	E
MAINTAINABILITY	G	G	E	E	G	G	A
VERIFIABILITY	E	G	G	E	G	E	G
EXPANDABILITY	E	A	E	E	E	A	G
FLEXIBILITY	A	G	G	E	A	G	G
INTEROPERABILITY	A	N/A	A	A	G	A	N/A
PORTABILITY	G	A	N/A	E	G	N/A	G
REUSABILITY	A	A	A	E	N/A	G	A

One problem with this goal specification process is that there is no relationship or quantification of the quality factors with respect to mission acceptability or performance. A System Project Officer does not know if "good" RELIABILITY is good enough, or if "excellent" is required. Baseline values or experience values would be a valuable addition to aid the SPO and provide him with a basis for picking a particular goal level.

Each decision aid is a relatively small system, and was therefore treated as a single Computer Software Configuration Item (CSCI). No subsystems were identified in the documentation for either aid, nor were separate sets of documents developed as they are for multiple CSCIs.

The software quality measurement (SQM) methodology directs that separate quality goals are to be developed for each system function. When the user surveys are distributed, however, no information is contained on the forms that specifies exactly what the identified functions are. As a result, it is up to each survey respondee to list the functions as he believes they exist. This can cause great confusion and difficulty. Section 4.3.9 contains SAIC's recommended solution to this problem.

As an example of the confusion that can occur, the four functions identified by the developer (PAR Technology) for ESCMA are:

- Driver
- Calculation
- Sensitivity Analysis
- Report Generation

However, the ESCMA functional description [PAR-4], describes the functions as:

- Identify areas of operation and aircraft/airfields of interest
- Update airfield and aircraft resource status
- Establish aircraft sortie and resource consumption rates and minimum requirements
- Develop objective function and constraint equations for optimization
- Compute maximum sortie rates
- Develop and document enemy sortie capability estimate

Functional decomposition is in general a subjective process, and typically results in varying lists of what constitutes the system functions. These discrepancies certainly were evident in this methodology step. For this reason, SAIC is recommending a modified approach to the user questionnaire. This approach is discussed in Sections 4.3.5 and 4.3.9, and uses functional goal specification only when functions have been identified and specified before quality goal specification is to take place. Otherwise, only system-wide goal setting is used.

SAIC did not use the functional allocation given by the developer or as shown in the documentation because we believe that quality results calculated on that basis are misleading. The subjectivity of functional decomposition detracts from the quantitative results that the software measurement technology is attempting to build and validate. We recommend that this additional subjectivity be eliminated until methods have been developed for more objective functional decomposition, and for the setting of various goals among functions based on sound reasoning and theoretical analyses.

To analyze the ESCMA and ECOAEA data, SAIC consolidated all of the various specified functional goals into single system-wide quality goals. To accomplish this, the highest functional goal for each factor was taken as the system-wide goal. As an example, the quality goals for the factor USABILITY were listed as Good, Excellent, Excellent, and Average for the four developer-identified functions. One approach to establishing a system-wide goal might be to average these functional goals (with a result of a goal of Good for USABILITY). We think it is better to have goals that reflect the highest standard desired, and so instead have chosen to establish the goal as Excellent. This was chosen because at this level, the quality can only be as good as its weakest function.

Table 3.3-1, presented earlier, lists the goals specified by the developer and by SAIC analysts. SAIC analysts were included in the survey for two reasons. The first was to give us experience in goal setting, to better allow assessment of how an acquisition manager or developer might respond to the survey. The second reason was to provide more data points for the process of final goal specification. The acquisition manager is likely to have several survey responses to consolidate, and we wished to duplicate his experience. The goals set by the SAIC analysts were derived to the best of their knowledge, but are not as meaningful as those that an acquisition manager would set himself. The goals listed in the table reflect the highest goal for any subsystem for each factor, as described above.

Using developer and SAIC survey responses resulted in correlating three separate sets of goals for the ESCMA aid, and four sets for the ECOAEA aid. Analysis was made of the variation in results for each factor for both aids.

For the ESCMA aid, the SAIC analysts agreed exactly on eight factors out of the thirteen. For four other factors, agreement was only one rating apart (between "Not Applicable" and "Average," or between "Good" and "Excellent," for example). One factor was two ratings apart ("Average" and "Excellent" for EXPANDABILITY). Including the developer's responses resulted in less agreement. Only three factors matched exactly, eight were one rating apart, and two were two ratings apart. This data, as well as data for the ECOAEA aid, are shown in Table 3.3-2.

TABLE 3.3-2 VARIATION IN GOALS AMONG ANALYSTS

VARIATION	ESCMA		ECAOAEA	
	SAIC	ALL*	SAIC	ALL*
NO VARIATION	8	3	2	2
1 LETTER GRADE	4	8	5	2
2 LETTER GRADES	1	2	6	5
3 LETTER GRADES**	0	0	0	4

* includes developer
** N/A is treated as one letter grade below average (A)

Analysis indicates that the differences in goal specification shifted toward greater variation when the goals of the developer were included. There was greater agreement among SAIC analysts than there was among SAIC analysts and the project developers. Less variations were also shown among the analysts for the ESCMA than for the ECOAEA. Since SAIC's metric-experienced team was assigned to ESCMA, these more consistent results were expected. These differences raise some questions about the subjectivity of this methodology, particularly in the case of the factors FLEXIBILITY and EFFICIENCY. The goals for these factors differed by as much as two steps.

Since SAIC analysts completed the survey form in order to collect data and provide multiple results for analysis efforts, the discrepancies between goal specifications are not meaningful as such. They do serve to re-emphasize, however, the subjective nature of the goal survey questionnaire and process. SAIC is recommending that efforts be made to lessen this subjective element with further research to validate scores against real-world, visible quality (see Section 4.2.2).

3.3.3 Select and Specify Quality Criteria

This SQM procedure involves selecting the criteria to be used in measuring the quality factors. The criteria for each applicable factor are selected and weighted to calculate the desired values.

Since the prototype aids do not access data from external data bases, the criterion of effectiveness of communication for the factor EFFICIENCY was considered to be inapplicable and was weighted as zero. Since neither decision aid communicates with any external systems, the criterion of system compatibility was also weighted as zero.

The criterion of effectiveness processing was considered to have greater impact on the factor EFFICIENCY than effectiveness storage, since storage is off-line and does not appear to be a limitation. The elimination of effectiveness-communication raised the weighting of each of the two remaining criteria from 33% to 50%. Considering effectiveness processing (EP) to be more important than the storage (ES) criterion resulted in raising the weighting of EP to 80% of the total, and the lowering of ES to 20%. These weightings, while reflecting our best estimation of what was appropriate, illustrate the arbitrary nature of the current criteria weighting methodology.

The impact of weighting the criterion system compatability as zero (since there is no inter-system communication), altered the weighting of the remaining criteria which determine the quality factor of INTEROPERABILITY, increasing each from 20% to 25%. All other criteria weights remain unaltered, since we had no justification for changing these weights. Table 3.3-3 indicates the composition of each of the quality factors by presenting their weighting formulas. In the table, some criteria weights are shown as decimal values, and others as fractional elements. Decimals were used whenever possible because they clearly represent the numerical values to be calculated. In some cases, however, fractions do not translate to finite decimal values (e.g., 1/3). For those cases, the fractions themselves are given in the table.

Only one particular problem was noted during criteria specification. The methods in the guidebook for selecting and weighting criteria are arbitrary, and without detailed justification. The acquisition manager has no way to link any assigned weightings to any real-world indicators or values. Since the scores are so dependent on this arbitrary assignment, there is no way for the manager to know that any calculated result reflects a real-world meaning (such as errors per thousand lines of code).

SAIC is recommending methods to help in the correlation of calculated quality scores and the actual quality of each system. One method, described in section 4.3.8, is to eliminate adjustment of criteria weighting. Instead, procedures would be devised to allow corrective efforts to focus on criteria of special concern. Section 2 also contains information on this problem, presenting data concerning the validity of the calculated decision aid quality scores.

3.3.4 Select and Specify Quality Metrics

Following the elimination of non-applicable quality factors and criteria (and the reweighting of criteria), the non-applicable individual metric elements were eliminated. The similarities between ESCMA and ECOAEA meant that the same weighting formulas and metric element questions could be applied to both aids. Table 3.3-4 shows the questions which were eliminated from each worksheet using this process.

TABLE 3.3-3 CRITERIA WEIGHTING FORMULAS

FACTOR	WEIGHTING FORMULA
EFFICIENCY	= 0(EC) + 0.8(EP) + 0.2(ES)
INTEGRITY	= NOT APPLICABLE
RELIABILITY	= [(AC) + (AM) + (SI)]/3
SURVIVABILITY	= NOT APPLICABLE
USABILITY	= 0.5(OP) + 0.5([illegible])
CORRECTNESS	= [(CO) + (CS) + (TC)]/3
[illegible]	[illegible]
[illegible]	[illegible] 25([illegible])
[illegible]	[illegible] (MO) [illegible]
[illegible]	[illegible] .25(SD) [illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible] (MO) + (SD) + (SI)]/9

TABLE 3.3-4

SQMD N/A QUESTIONS

WORKSHEET 1	WORKSHEET 2	WORKSHEET 3	WORKSHEET 4

AM.1(1)
AM.6(1)
AM.7(1) - (3)
AT.3(2)
AU.2(1)
CL.1(1) - (12)
CL.2(1)
CL.2(3) - (8)
CL.3(1)
CP.1(11)
CS.2(4) - (5)
DI.1(2)
DI.1(4)
DI.1(6) - (9)
EC.1(1)
FO.1(1) - (4)
FS.2(2)
FS.2(6)
ID.1(2) - (3)
MO.2(1) - (5)
OP.1(14)
RE.1(1)
RE.1(3)
SD.3(5)
SS.1(4)
SY(ALL)
	AM.6(1) - (4)
	AM.7(1) - (3)
	AT.3(2)
	CL.1(2) - (8)
	CL.1(11)
	CL.2(2)
	CL.2(4)
	CL.2(6)
	CP.1(11)
	CS.2(4)
	CS.2(5)
	DI.1(4)
	DI.1(6) - (9)
	EP.1(5)
	EP.2(3)
	MO.1(9)
	MO.2(3)
	MO.2(5)
	OP.1(4)
	OP.2(6)
	RE.1(1)
	RE.1(3) - (4)
	SS.1(1) - (4)
	SS.2(1) - (2)
	SY(ALL)
		AT.3(1) - (2)
		AU.1(2)
		CL.1(7) - (8)
		CL.2(1)
		CP.1(11)
		ES.1(4)
		OP.1(10)
			AM.2(2)
			AM.2(4)
			AT.3(1) - (2)
			CL.1(7) - (8)
			CL.2(1)
			CP.1(11)
			CS.1(2) - (4)
			CS.2(1) - (3)
			EP.1(5)
			ES.1(4)
			ES.1(7)
			OP.1(1)
			OP.1(2)
			SD.2(1)
			SD.2(2)
			SD.2(4)
			SD.2(5)
			SD.3(5)
			SI.4(13)
			SI.4(14)
			VS.1(1) -(2)

PP/RD/01

In a standard acquisition process, the elimination of metric questions would mean that they were, of course, not scored. Because of the research nature of this effort, however, all questions were evaluated and scored. This resulted in two scores for quality assessment: one reflecting only those elements that were not eliminated, and one reflecting all question answers.

Because individual metric questions are not weighted, and are eliminated only on the basis of criteria and factor elimination, we did not uncover a problem with the arbitrary nature of scoring or removing questions. We did have difficulty, however, with the weight that each question receives. Section 4.2.2 discusses this as an area for continued research. An example of the problems in this area is in the evaluation of the aid ECOAFA during the code phase. The factor VISIBILITY received a score of .98 for that phase. That entire score was based on answering one question, OP.1(10). All other questions were not applicable (there are three operability questions in that phase).

In addition, we found no methodological support for the user when he is driven by cost reasons to measure only some aspects of the system. No guidance existed as to how to pick out the factors/criteria/metric elements that provided the most cost-effective means of reaching his desired quality goals. SAIC presents a recommended solution for this problem in Section 4.3.4.

3.4 Metric Application

Once quality goals have been defined and specified, the products of the software development process must be evaluated for compliance to these goals. Volume III of the software quality measurement guidebooks [8, 3] includes instructions for the evaluation of achieved software quality. This evaluation is defined as data collection, data analysis, and generation of the software quality evaluation report. All these elements are then delivered to the System Program Office so that they may easily assess project compliance.

This section briefly describes the data collection aspect of assessing software quality. Section 3.5 describes scoring, and Section 3.6 discusses the quality evaluation reports. Only problems encountered that pertain to the metric methodology are described here; all technical problems uncovered concerning the decision aids are discussed in Section 2.

The SQM methodology depicts the general flow for evaluating achieved quality levels as in Figure 3.4-1. It states that development products are used as source material for answering questions on the metric worksheets. Answers on the worksheets are used to score metric elements on scoresheets, and scores are calculated for the parent metrics, criteria, and factors. Scoring results are compared to requirements and variations analyzed. All results are documented in a Software Quality

Evaluation Report and submitted to the System Program Office.

The steps in the procedure for scoring quality aspects are to first identify allocation relationships, and then to apply the worksheets. SAIC's approach to this process was to follow the steps identified in the SQM guidebooks as closely as possible, while keeping detailed records on results, time spent, and material covered.

The allocation identification process described in the guidebooks did not lend itself well to this application. Discussion of this process is listed below in Section 3.4.1.

Worksheet application was not described in detail in the methodology. SAIC performed this task by creating one team to score each decision aid. The team scoring the ECOAEA aid primarily consisted of analysts who were inexperienced in metrics. The team scoring the ESCMA aid primarily consisted of analysts experienced in metrics. The following assumptions and techniques were used to score the worksheets:

- All metric elements on each worksheet were scored. In a standard application of the methodology, only those metric elements which are determined to be applicable during quality goal specification would be scored. Because of the research nature of this project, however, every question was to be answered and associated data collected.

- Time spent for each metric element was tracked on the project time log sheets shown earlier. Time spent, amount of material covered, and the identification numbers of Technical Problem Reports and Methodology Problem Reports created were recorded.

- Each analyst was assigned an arbitrary section of the worksheet to complete for Worksheets 0, 1, 2, and 3A & 3B. For Worksheet 4A & 4B, questions were allocated based on analyst experience. Some questions are difficult to answer for non-programmers. Section 4.4.1 discusses this further.

- The metric element questions were completed in order as they appeared on the worksheet.

- If the score on a metric element (for Worksheets 0, 1, and 2) was neither "YES" nor "1", a Technical Problem Report was completed. This report describes a metric violation. For Worksheets 3B and 4B, no problem reports were to be completed -- instead, we had planned to produce the reports based on the scoring onto Worksheets 3A and 4A. Time did not permit the creation of problem reports on those worksheets. Since, however, most questions are covered in the early phases of development, this did not create a problem for analysis.

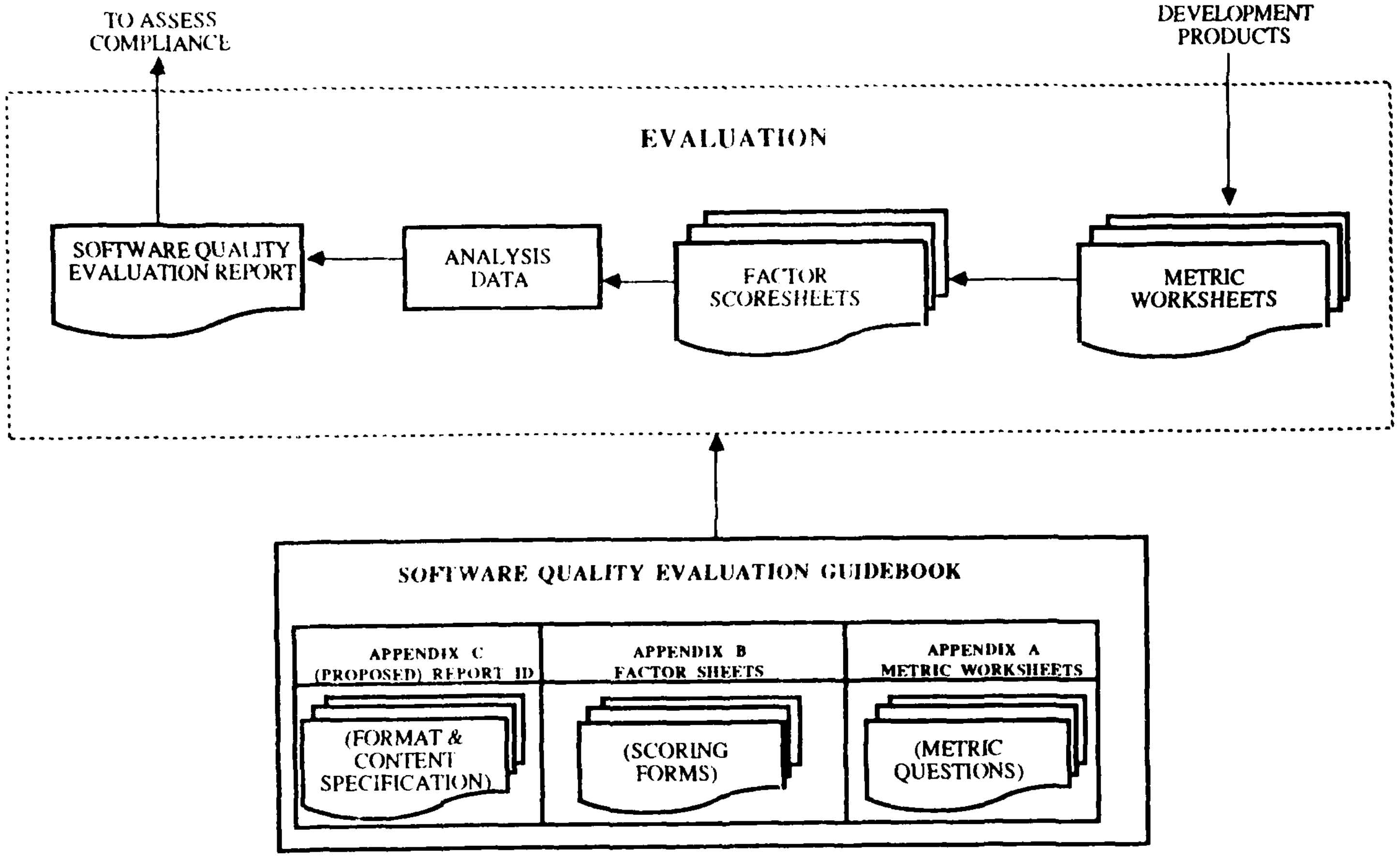

TO ASSESS COMPLIANCE
DEVELOPMENT PRODUCTS
EVALUATION
SOFTWARE QUALITY EVALUATION REPORT
ANALYSIS DATA
FACTOR SCORESHEETS
METRIC WORKSHEETS
SOFTWARE QUALITY EVALUATION GUIDEBOOK
APPENDIX C (PROPOSED) REPORT ID
APPENDIX B FACTOR SHEETS
APPENDIX A METRIC WORKSHEETS
(FORMAT & CONTENT SPECIFICATION)
(SCORING FORMS)
(METRIC QUESTIONS)

FIGURE 3.4-1 METRIC DATA COLLECTION, ANALYSIS, AND REPORTING

- If any questions or difficulties arose during metric element evaluation, a Methodology Problem Report was to be completed.

These steps and assumptions were used to evaluate all the worksheets. Since each decision aid consisted of a single Computer Software Configuration Item (CSCI), the differentiation between the system-level Worksheet 0 and the CSCI-level Worksheet 1 was not large. Because of this, no new information would be gained by evaluating both Worksheets 0 and 1 for each aid. For research purposes, we decided to evaluate Worksheet 0 at the system level for all of the decision aids developed. We used the decision aid statement of work (SOW) and the planning document [PAR-2] to make this evaluation. Worksheets 1 through 4A & 4B were evaluated for both aids. Table 3.4-1 lists the documents used for the worksheets evaluated for each aid.

These documents were selected as the best fit available for the intent of the evaluation process. This evaluation was done outside of the system development process and after test project completion. This means that all results were gathered after the full implementation of the aids, and therefore could not influence the development process.

It was PAR's intention to develop the decision aids much in accordance with MIL-STD-7935.1-S, Automated Data Systems Documentation Standards [7935], but in fact the documents and life cycle do not entirely follow this standard. This was due to the prototype nature of the decision aid project.

Even though the project did not entirely correspond to the standards of MIL-STD-7935.1-S, it is important to include the standard in order to understand the developer's intent. Documents required for MIL-STD-7935 development can be assigned to two basic categories: system documents and collateral documents. System documents are those engineering documents used to define, build, and maintain the system. Collateral documents include those that manage and control the development process, that provide standards, that describe how to use it, and report on testing. The reason for this decomposition is to provide a simplified frame-of-reference for comparison to another military standard, DOD-STD-2167. Documents in each category for MIL-STD-7935.1-S are listed below:

COLLATERAL

- User's Manual
- Computer Operation Manual
- Program Maintenance Manual
- Test Plan
- Test Analysis Report

SYSTEM

- Functional Description
- Data Requirements Document
- Program Specification
- Data Base Specification

TABLE 3.4-1 DOCUMENTS EVALUATED

WORKSHEET	AID	DOCUMENT
0	SYSTEM	INTERIM TECHNICAL REPORT. SENIOR BATTLE STAFF DECISION AIDS. TASK 1: PLANNING (3-84) [PAR-2] STATEMENT OF WORK FOR SENIOR BATTLE STAFF DECISION AID (12/82) [PAR-3]
1	ECOAEA	ENEMY COURSE OF ACTION EVALUATION AID FUNCTIONAL DESCRIPTION AND DESIGN PLAN. SENIOR BATTLE STAFF DECISION AIDS (7-84) [PAR-5]
1	ESCMA	ENEMY SORTIE CAPABILITY MEASUREMENT AID: DESIGN PLAN AND FUNCTIONAL DESCRIPTION (8/85) [PAR-4]
2,3	ECOAEA	ENEMY COURSE OF ACTION EVALUATION AID FINAL FUNCTIONAL DESCRIPTION AND DESIGN PLAN (10/85) [PAR-6]
2,3	ESCMA	ENEMY SORTIE CAPABILITY MEASUREMENT AID: DESIGN PLAN AND FUNCTIONAL DESCRIPTION (8/85) [PAR-4]
4	ECOAEA	SOURCE CODE LISTINGS
4	ESCMA	SOURCE CODE LISTINGS

- System/Subsystem Specification

A draft copy of the DOD-STD-2167 standard, Defense System Software Development, was the basis for the guidebook's software quality measurement technology. In order to make comparison to MIL-STD-7935.1-S easier, the documents described in that standard have also been divided into two types, system and collateral:

COLLATERAL

- Operational Concept Doc.
- Software Configuration Mangement Plan
- Software Quality Evaluation Plan
- Software Development Plan
- Software Test Description
- Software Test Plan
- Computer Resources Integrated Support Document
- Computer System Operator's Manual
- Software User's Manual
- Computer System Diagnostic Manual
- Software Programmer's Manual
- Software Test Report
- Firmware Support Manual
- Version Description Document
- Software Test Procedure

SYSTEM

- System/Segment Spec.
- Interface Requirements Specification
- Software Requirements Specification
- Software Top Level Design Document
- Interface Design Document
- Data Base Design Document
- Software Detailed Design Document
- Source Code (Software Product Specification)

Comparison between the documents required for each standard yielded a correspondence as shown in Figure 3.4-2. Given this framework, it was necessary to analyze the existing and delivered system documentation against both MIL-STD-7935.1-S and DOD-STD-2167. This was required to understand the documents the guidebooks were designed to use, as well as the documents actually to be used for evaluation. Tables 3.4-2 and 3.4-3 present the documents available for analysis for each decision aid. Figure 3.4-3 is a synthesis of the preceding figures and tables and represents the documents available and their relation to the desirable set of specifications.

A later version of the functional description was used for scoring Worksheets 2 through 4 for both decision aids. These specifications became available after the scoring of Worksheet 1 was completed.

Analysis of the methodology problems uncovered during the scoring of these worksheets is contained in the following sections.

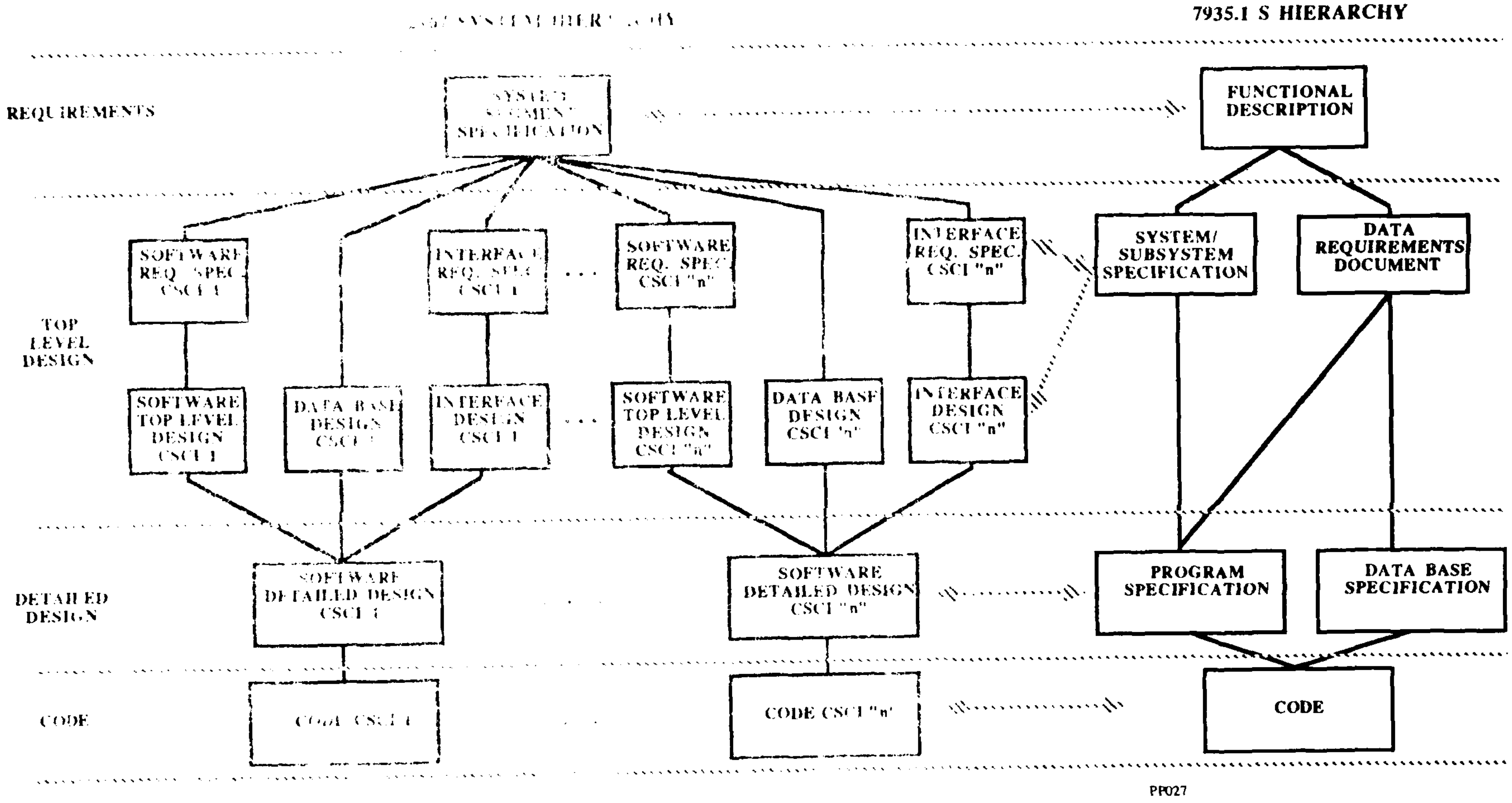

SYSTEM HIERARCHY
7935.1 S HIERARCHY
REQUIREMENTS
TOP LEVEL DESIGN
DETAILED DESIGN
CODE
SYSTEM/ SEGMENT SPECIFICATION
SOFTWARE REQ. SPEC CSCI 1
DATA BASE DESIGN CSCI 1
INTERFACE REQ. SPEC CSCI 1
SOFTWARE REQ. SPEC CSCI "n"
INTERFACE REQ. SPEC. CSCI "n"
SOFTWARE TOP LEVEL DESIGN CSCI 1
INTERFACE DESIGN CSCI 1
SOFTWARE TOP LEVEL DESIGN CSCI "n"
DATA BASE DESIGN CSCI "n"
INTERFACE DESIGN CSCI "n"
SOFTWARE DETAILED DESIGN CSCI 1
SOFTWARE DETAILED DESIGN CSCI "n"
CODE CSCI 1
CODE CSCI "n"
FUNCTIONAL DESCRIPTION
SYSTEM/ SUBSYSTEM SPECIFICATION
DATA REQUIREMENTS DOCUMENT
PROGRAM SPECIFICATION
DATA BASE SPECIFICATION
CODE
PP027

FIGURE 3.4-2 SYSTEM DOCUMENTATION

TABLE 3.4-2 ECOAEA DOCUMENTATION

DOCUMENT	DESCRIPTION	7935.1-S "SYSTEM" APPLICABILITY
Functional Description Design Plan (10/85)	Requirements, High-Level Design, PDL, Data Dictionary	Functional Description, System Specification, Program Specification*
Computer Program Development Plan (10/85)	Development Organization and Methodology, WBS, Schedule	N/A
Test Evaluation Plan	Not Software Testing, but Evaluation of ECOAEA Concept	N/A
Demonstration Plan Draft 10/85	Plan On How to Demo System	N/A
Task 1: Planning Document	General Requirements, Project Plan	Some Content Like Functional Description
Selected Proposal Pages	Description of Project Plans	N/A
Various	Information on Aids and General SBSDA Briefings	N/A
Code		Code

*Contents similar, but not high-quality match.

TABLE 3.4-3 ESCMA DOCUMENTATION

DOCUMENT	DESCRIPTION	7935.1-S "SYSTEM" APPLICABILITY
Computer Program Development Plan (9/85)	Project Plan, Staffing, Tasks, WBS, Schedule, Concept of Operations, Development Methodology	N/A
Test Evaluation Plan (1/85)	Evaluation of ESCMA Concept, not Software Testing	N/A
Design Plan and Functional Description (9/85)	Requirements, Preliminary Design, Data Dictionary	Functional Description, System Specification, Program Specification*
Test/Evaluation Analysis Report (10/85)	Results of Test Evaluation Plan	N/A
Test/Demonstration Plan (7/85) Draft	5 Volumes. Plan to Demo ESCMA. Scenarios. User Guide.	N/A
Task 1 Planning Document (9/83)	Description of All Aids, Project	Somewhat has FD-Type Material
Selected Proposal Pages	Information On Project	N/A
Various	Assorted Briefings, Discussions	N/A
Code		Code

*Contents similar, but not high quality match.

DECISION AID "SYSTEM" DOCUMENTATION
(Excluding Test Documents)

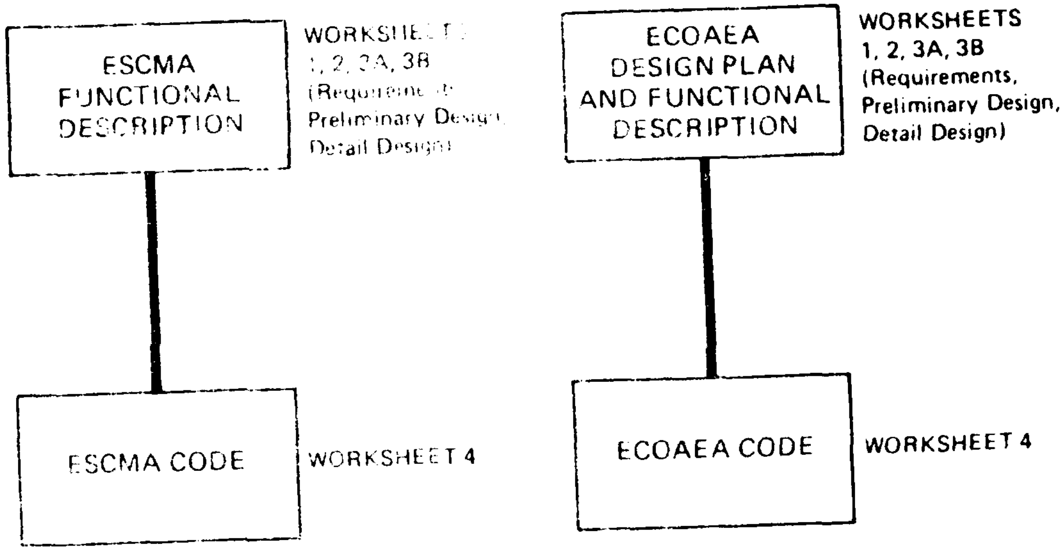

ESCMA FUNCTIONAL DESCRIPTION
WORKSHEETS 1, 2, 3A, 3B (Requirements, Preliminary Design, Detail Design)
ECOAEA DESIGN PLAN AND FUNCTIONAL DESCRIPTION
WORKSHEETS 1, 2, 3A, 3B (Requirements, Preliminary Design, Detail Design)
ESCMA CODE
WORKSHEET 4
ECOAEA CODE
WORKSHEET 4

FIGURE 3.4-3 DECISION AID "SYSTEM" DOCUMENTATION

3.4.1 Identify Allocation Relationships

The ESCMA and ECQAEA decision aids are each relatively small systems, and were not decomposed into Computer Software Configuration Items (CSCIs) or Computer Software Components (CSCs) during the development process. As discussed in paragraph 3.3.2, there were also discrepancies in the functions specified in the documentation of each system and as listed by the developers in the user survey questionnaire. These discrepancies illustrate the subjectivity inherent in functional decomposition, and the variations that exist among the resulting functional lists. For these reasons, SAIC did not allocate separate quality requirements among a list of system functions for either aid. Instead, both aids were given system-wide goals based on developer and SAIC team responses to the user survey questionnaire.

Because of that selection, it was not necessary to identify the relationship between the specified functions and the system documentation as developed. We still examined the methodology, however, in order to suggest improvements and assess its feasibility.

We believe that the identification process has difficulties. Because functional decomposition is subjective and arbitrary, the allocation of requirements to CSCI's based on these functions is also arbitrary. The assessment of the amount of a particular CSCI that reflects a particular function is also difficult and subjective. For these reasons, we are recommending that, in general, no functional decomposition and allocation of requirements be contained in the current guidebooks. As research proceeds and these types of relationships are verified, the procedures should be placed back in the guidebooks with full justification for their existence and with definite steps and guidance as to how to perform this task.

For systems that do merit functional decomposition (as described in paragraph 4.3.5), we recommend that CSCIs be allocated to functions (rather than the reverse). This means that rather than trying to determine how much of a particular CSCI (e.g., 50%) relates to a particular function, one would determine whether or not a CSCI aids in implementing a function at all. If it does, then the CSCI would be counted 100% in the score created for that function. As an example, consider the example of calculating optimal values as a function. For this function, any CSCI that performed any part of the task would be considered in the scoring. One would not take into account that only 10% of a given CSCI actually contributed to that particular function.

Section 4.3.5 contains other recommendations applicable to identifying the allocation of functional requirements among the CSCIs.

3.4.2 Application of Worksheets

A major aspect of the analysis of the SQM methodology is evaluating the metric element questions that comprise the worksheets. This analysis of the SQM methodology metric element questions has been done for all worksheets.

Several general statements can be made about the organization of the worksheets. These statements are explained in detail below, along with recommendations for correcting deficiencies.

Each worksheet is organized with the questions in alphabetical order by metric mnemonic, but the data needed to answer the metric questions have no relationship to this alphabetical order. As a result, the same question may appear more than once on each worksheet. In addition, many of the questions (discussed below in detail) were [illegible] [illegible] or confusing. There is also a lack of information concerning [illegible] sources for question answers and a lack of adequate definitions of terms used. In some cases, we found high level and low level [illegible] to be found on the same worksheet.

The worksheets require analysts [illegible] [illegible] [illegible] [illegible] mechanism for documenting [illegible] technical problems identified. For example, metric element [illegible] [illegible] [illegible] of all references [illegible] been defined. The process [illegible] [illegible] listing [illegible] [illegible], [illegible] all referenced functions, and comparing the two. The evaluation [illegible] notes, determines the answers and responds [illegible] the worksheet. There [illegible] no mechanism for recording this data. Updates to documents [illegible] require full re-evaluation because of this.

In other cases, questions are not answerable. If the question [illegible] an unknown value, the analyst must eliminate the question (and thereby increase the weightings of remaining questions), enter artificial values to generate a score of zero or one, or enter N/A when that is not appropriate. Questions which require values do not allow for their lack. An example of this type of question is [illegible] on worksheet [illegible]. This question concerns the number of overlays used in the [illegible]. If overlays are not discussed in the top level design documentation, how can the analyst score this question? It may be that no overlays are to be used (for an N/A score) or it may be that they are to be used but are undocumented (for a score of 0).

3.4.2.1 Metric Comment Categories

The worksheet evaluation task involved answering 327 metric elements. with 850 questions appearing on the 5 worksheets. Of the metric elements, 153 (47%) received problem comments of one form or another. Of the questions comprising these elements, 317 (37%) received problem comments.

The types of problems encountered during worksheet evaluation can be grouped into nine categories, with many of the metrics commented upon for more than one problem. These categories are listed in earlier in Table 3.4-4. The following paragraphs describe each of the categories and provide examples of each problem.

TABLE 3.4-4 METRIC PROBLEM CATEGORIES

CATEGORY	PROBLEM
1	Question is inappropriate for the worksheet level.
2	Question is confusing.
3	Question generated a miscellaneous comment (e.g. decompositio needed, question dependent on previous worksheet).
4	Question is too subjective.
5	Question requires an "all or nothing" response.
6	Question contains a typographical error, or some other format problem.
7	Question is a duplicate of another question on same worksheet.
8	Calculating the question's score involves dividing by zero, or there is some other scoring difficulty.
9	Question should be in a block of questions that are nested by topic.

Category 1: The question is inappropriate for this worksheet level. Some questions were found by the analysts to be inappropriate for the phase or level under analysis. For example, one would not expect to see data flow diagrams and discussion of control flow in high-level requirements documents. The example question below, however, specifically calls for this information. The Data Item Descriptions for DOD-STD-2167 do not call for this information at the requirements level. The System/Segment Specification (DI-CMAN-80008), for example, does not contain a requirement for this type of data. It does require, however, that functional flow be shown. This higher level of detail (functional flow as opposed to control flow) is expected to be in system requirements documentation. For each of these questions, SAIC recommends removing them from the applicable worksheet. An example of this type of question is:

> Worksheet 0, Question IM.2(3). Does the requirements/design documentation [illegible] control and data flow (e.g., graphically portrayed with [illegible] explanations or DFD)?

Paragraph 4.4.9 discusses SAIC's recommended solution to this problem. For each of these questions, we recommend that the question be eliminated from the inappropriate worksheet.

Category 2: The question is confusing. This category was indicated for more metric element questions than any other category, except for Category 8. Some questions contained terms that were either poorly defined in the glossary, or not defined at all. Other questions could be understood term by term, but examples would be necessary to explain the concept. Some of the confusion could be blamed on the glossaries, which were sparse and identical for each worksheet. Some terms need to be defined differently for each worksheet (e.g., function and system). In one case two different terms were used in different contexts but had the same definition. An example of this type of problem, with such terms as "specific data storage and retrieval references" unclear, is:

> Worksheet 3B, Question AP.2(3). Is the logical processing free from specific data storage and retrieval references (e.g., data symbolically defined and referenced)?

SAIC's recommended solution for this problem is found in paragraphs 4.4.6, 4.4.7, 4.4.8, and 4.2.3. We recommend training be supplied as an aid to standard worksheet evaluation, and that a glossary, metric examples, and standard procedures be developed.

Category 3: The question generated a miscellaneous comment. This category includes questions that could be decomposed, that were dependent on answers supplied in a previous worksheet, that should be moved from a unit-level worksheet to a CSCI-level worksheet, and that contained content with which we disagreed. Compound questions ask one question about several different things and become difficult to answer when not all the answers are the same. Complex questions force you to answer several questions before you have enough information with which to answer the "real" question. An example of a compound question is:

> Worksheet 1, Question TN.1(1). Are there requirements to provide lesson plans and training materials for operators, end users, and maintainers of the CSCI?

Some questions are asked for each unit, but we recommend that they only be asked once for the complete CSCI. An example of this is question AM.2(7) for Worksheet 3B. This question implies that each unit should check for all data. We recommend moving the question to Worksheet 3A, and changing it to read: "Is a check performed to determine that the data used by all units is available before it is used in processing?" The original Worksheet 3B question reads:

> Worksheet 3B, Question AM.2(7). Is a check performed before processing begins to determine that all data is available?

Since we required analysts to answer all questions (even if not applicable to the decision aids), they noted several questions which were dependent on questions answered in prior worksheets. The consistency questions are an excellent example of this problem. Early questions concern the existence of standards, while later worksheet questions relate to complying with these standards. We believe it would greatly help the worksheet tailoring process if lists of these related questions were made available to ensure that they are consistently tailored out. An example of this relationship is:

> Worksheet 1, Question CS.1(1). Have specific standards been established for design representations (e.g., HIPO charts, program design language, flow charts, data flow diagrams)?

> Worksheet 2, Question CS.1(1). Are the design representations in the formats of the established standard?

Some questions contained content with which we disagreed. An example of this is question AU.1(2) on Worksheet 3A. This question requires estimating source code lines at the design level. We believe this question can give misleading results, as the estimation of lines of source code is very subjective. The question reads:

> Worksheet 3A, Question AU.1(2). How man estimated lines of source code, excluding comments? ...

Category 4: The question is too subjective. These questions contain qualitative adverbs like "minimally", "typically", "completely", "clearly", and "precisely". The answers to these questions depend totally on the analyst's subjectivity, and may not be an accurate reflection of the software being evaluated. The subjective quality of these questions also limits the value of comparing scores for products that were measured by different analysts. An example of this type of question is:

> Worksheet 2, Question AU.1(1). Are all processes and functions partitioned to be logically complete and self contained so as to minimize interface complexity?

By supplying guidance and procedures for these types of question, some of the subjectivity can be removed. In addition, we recommend that certain questions be answered by more experienced analysts -- these subjective questions fall into this category. These recommendations are discussed in paragraphs 4.4.1, 4.4.6, 4.4.7, and 4.4.8. Such questions as these are difficult to automate, unlike more countable elements like lines of source code. It is not the goal of this process, however, to eliminate experienced and competent experts from the evaluation process. Rather, the focus is to automate questions that do not require a great deal of judgment, and supply analysts with the information needed to answer the remaining questions in an effective and efficient manner.

Category 5: The question requires an "all or nothing" response. This category consists of questions that have the word "all" in them. The question is both difficult to answer and potentially misleading. The use of "all" leaves the scope of the question up to the particular analyst answering the question. For example, if the question concerns "all hardware errors", can the analyst be satisfied with answering the question regarding all the hardware errors mentioned in the documentation, or must the analyst consider all the hardware errors that could possibly occur? The answers to these questions may not accurately reflect the true state of the product/document, because the instance where only 1 out of 1000 fails is scored the same way as the instance where 999 out of 1000 fail. An example of this kind of question is:

> Worksheet 1, Question CP.1(1). Are all inputs, processing, and outputs clearly and precisely defined?

In general, SAIC recommends that these questions be retained as written for now. This is discussed in paragraph 4.4.10. Procedures can also help solve this problem, as shown in paragraph 4.4.7. The data collection workbook described in paragraph 4.4.1 could also be used as a partial solution to this difficulty. The workbook could be used to record the analyst's assessment of the severity of an "all or nothing" failure. The analyst would need to be fully qualified to make such judgments (i.e., of a certain experience level), and this is also discussed in paragraph 4.4.1.

Category 6: The question contains a typographical error, or some other format problem. Some questions contain typographical errors, or have a problem with the format in which they are presented. While this category does not cause major problems in general, it is an area that needs correction. As engineers at PAR Technology noted, the software quality measurement process should itself be of high quality. These sorts of minor errors may cause the software community to perceive software quality measurement in an unfavorable light. An example of this sort of problem is:

Worksheet 3A, Question SI.6(1), Part d. How many unique operations?

The word "operations" should read "operators." This and similar problems are discussed with recommended solutions in paragraph 4.4.3.

Category 7: The question is a duplicate of another question on the same worksheet. Some questions appear on the worksheet, with applicability to different criteria, more than once. This is not an error and does not imply that there is something wrong with the methodology, but it is an extremely inefficient data collection technique. Examples of this are shown by:

Worksheet 1, SI.1(3). Are there requirements for a programming standard?

Worksheet 1, SI.1(12). Are there requirements for a programming standard?

This problem can be solved by using a methodical data collection approach. Paragraph 4.4.1 discusses this solution.

Category 8: Calculating the question's score may involve dividing by zero, or there is some other scoring difficulty. For some questions, calculation of this step may result in dividing a number by zero. There are [illegible] that [illegible]. Often, the question is [illegible] answer, [illegible] for the first part would [illegible] the [illegible]. An example of this type of [illegible]:

Worksheet [illegible] Question [illegible]
[illegible]
[illegible] entered [illegible].

[illegible] that [illegible] allow the [illegible] questions that [illegible] manner. [illegible] question cannot be [illegible] it is [illegible] answer this question. An example of this situation is:

Worksheet 4, Question EP.1[illegible]. How many different overlays are used in this CSCI?

If the documentation does not mention or discuss overlays in any way, this question is hard to answer. It may be that no overlays exist, or it may be that overlay discussion was omitted when it should have been included.

Paragraphs 4.4.7 contains SAIC's proposed solution to this problem, which is to provide guidance for the analyst as to how to score a question in these situations.

Category 9: The question should be in a block that is nested by topic. For these questions, an answer to an earlier question can eliminate the need to answer subsequent questions. In the example below, a "no" or "N/A" answer to question CL.1(2) means that the subsequent questions should also receive a "no" or "N/A" response. There should be no need to spend extra time re-evaluating the subsequent questions.

> Worksheet 1, Question CL.1(2). Is there a requirement for a protocol standard to control all network communications?
>
> Question CL.1(3). Is the network processing control part of the network protocol standard?
>
> Question CL.1(4). Is user session control part of the network protocol standard?
>
> Question CL.1(5). Is communication routing part of the network protocol standard?
>
> Question CL.1(6). Is uniform message handling (e.g. synchronization, message decoding) part of the network protocol standard?

The use of a formal data collection workbook, described in paragraph 4.4.1, is SAIC's recommended solution to this type of problem.

3.4.2.2 Distribution of Comments

This section deals with the number of metric elements reported as having problems and the relationship to the number of elements comprising the criteria and factors. This information is calculated separately for each worksheet, but not for each aid. For the purposes of evaluating the methodology, the aid being analyzed when a metric problem was encountered is not meaningful.

There are a total of 73 metrics that have their metric element questions asked throughout the 5 worksheets. These metrics are composed of a total of 327 metric elements, with 850 questions asked in all 5 worksheets. Of these 850 questions, 317 (37%) were reported as having problems by SAIC analysts. Of the 327 metric elements, 153 (or 47%) were reported as having one or more problems.

These problems were grouped into the nine categories discussed above. Figure 3.4-3a shows each of these nine categories. The figure also reflects the relative number of metric elements reported for each of the categories. As an example, 19% of the metric element comments fall into the category of "All or Nothing" comments (see Category 5 above). Not only were varying numbers of metric elements reported in each category, but the categories themselves are of unequal criticality.

Some of the categories are in the nature of formatting problems. This means that they concern typographical errors, duplication of questions, questions which could be nested, and other minor problems. These questions can be easily solved by reformatting the worksheets. This includes Categories 6, 7, and 9 (errors, duplication, and nesting level), and represents 17% of the problems uncovered.

Other questions are more of a procedural nature. Unscorable questions, for example, can be handled by instituting procedures to guide analysts. This includes Categories 4 and 8 (subjective and unscorable), and represents 31% of the problems uncovered.

The final, and we believe most important grouping of categories, is more of a content problem. These Categories are 1, 2, and 5 (inappropriate level, confusing, and all or nothing). These problems require changes in the questions themselves, and comprise 50% of the problems uncovered. (The remaining 2% of the problems identified, in the miscellaneous category, were not allocated to any of these three groupings.)

The difficulty in correcting these groupings of categories will vary. Formatting problems are relatively easy to correct, and will not likely generate much controversy or discussion. Procedural problems are somewhat easily corrected, and can probably be accomplished without a great deal of discussion. The content problems, however, will be harder to correct without a great deal of discussion and information interchange among the Government agencies and contractors currently active in the metric community. Future contracts and technical interchange will help to solve this problem.

The metric elements combine to form a total of 29 criteria. Several of the criteria were not reported for any problems on any worksheet. These criteria are distributedness, effectiveness communication, effectiveness storage, generality, system accessibility, system compatability, tracebility, and visibility. The remaining 22 criteria were reported for between 15% and 100% of their metric element questions. The criteria accuracy, functional overlap, and virtuality had the highest percentage of metric element questions reported with 100%. Figure 3.4-4 displays this data for all criteria.

Figure 3.4-5 illustrates the percentage of the comments on each criterion, relative to the worksheet involved. For each criterion, the figure shows the percentage of metric elements reported out of the total present on each of the five worksheets.

The criteria are combined to form thirteen factors. Only the factor INTEGRITY did not have any metric element questions reported. The

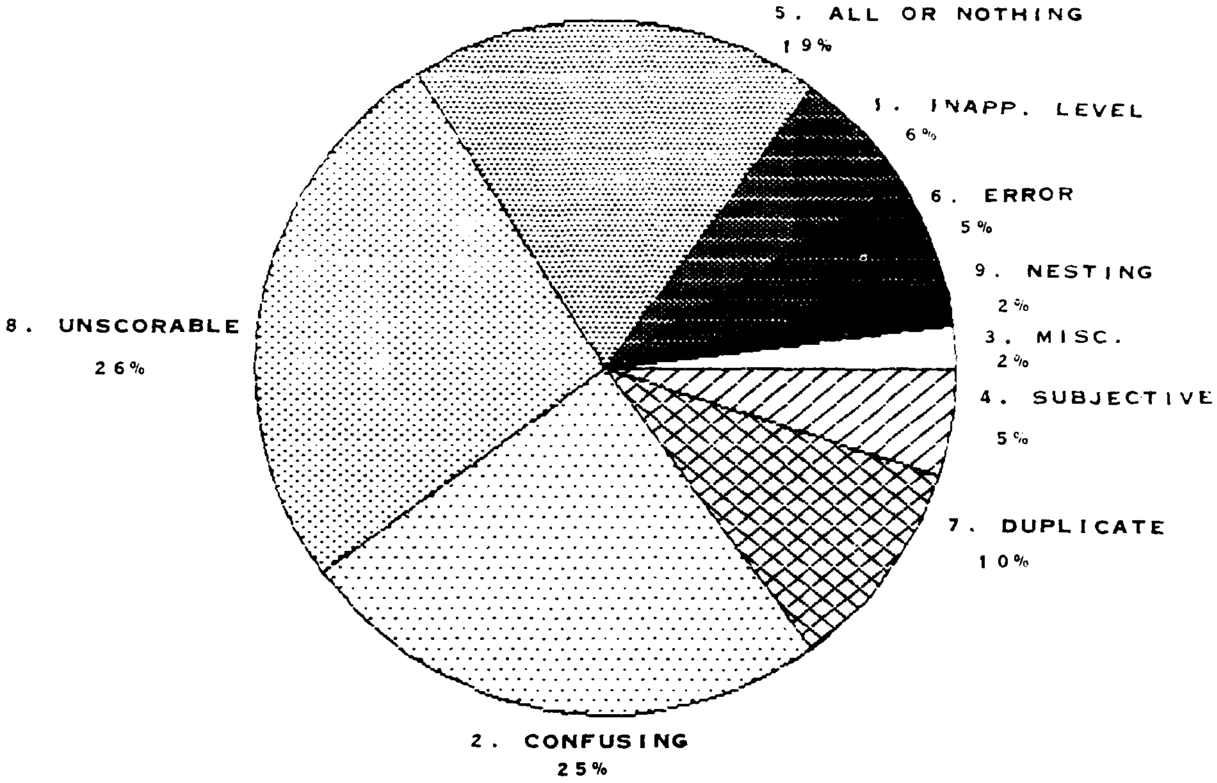


Figure 3.4-3a Distribution of Comment Types

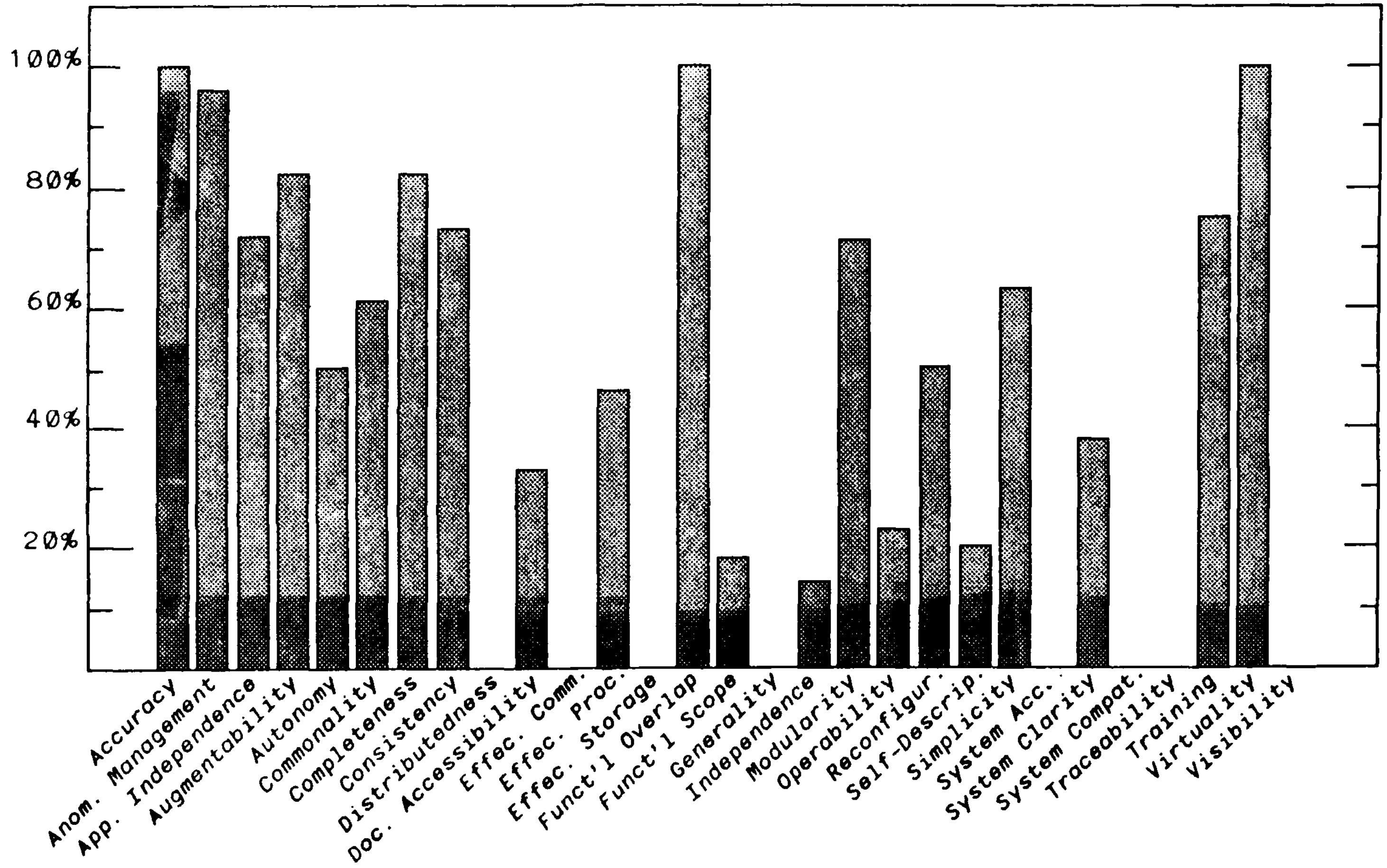


Figure 3.4-4
Percentage of Metric Elements Commented for Each Criterion

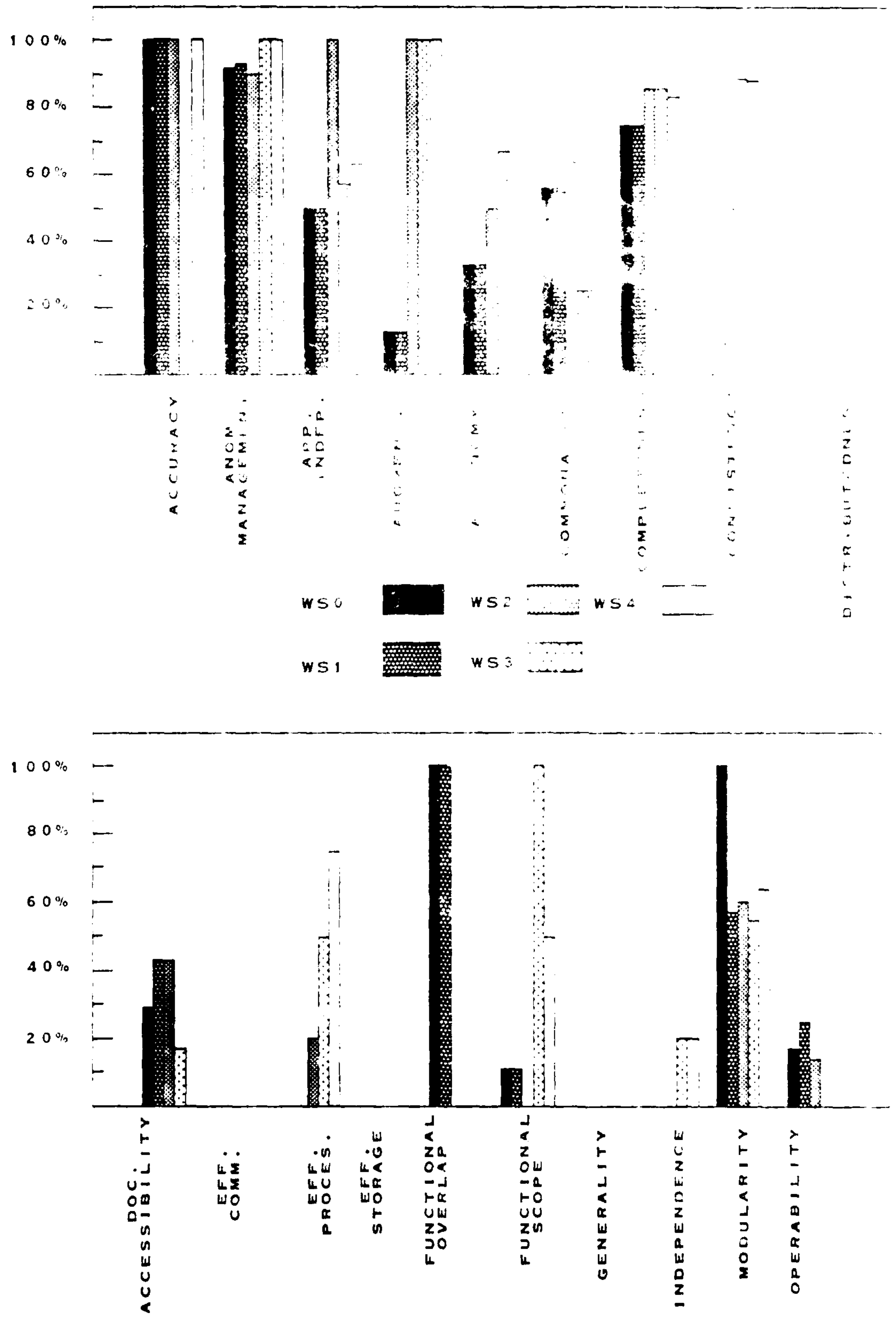


Figure 3.4-5 Percentage of Comments on Metric Elements Across All Worksheets

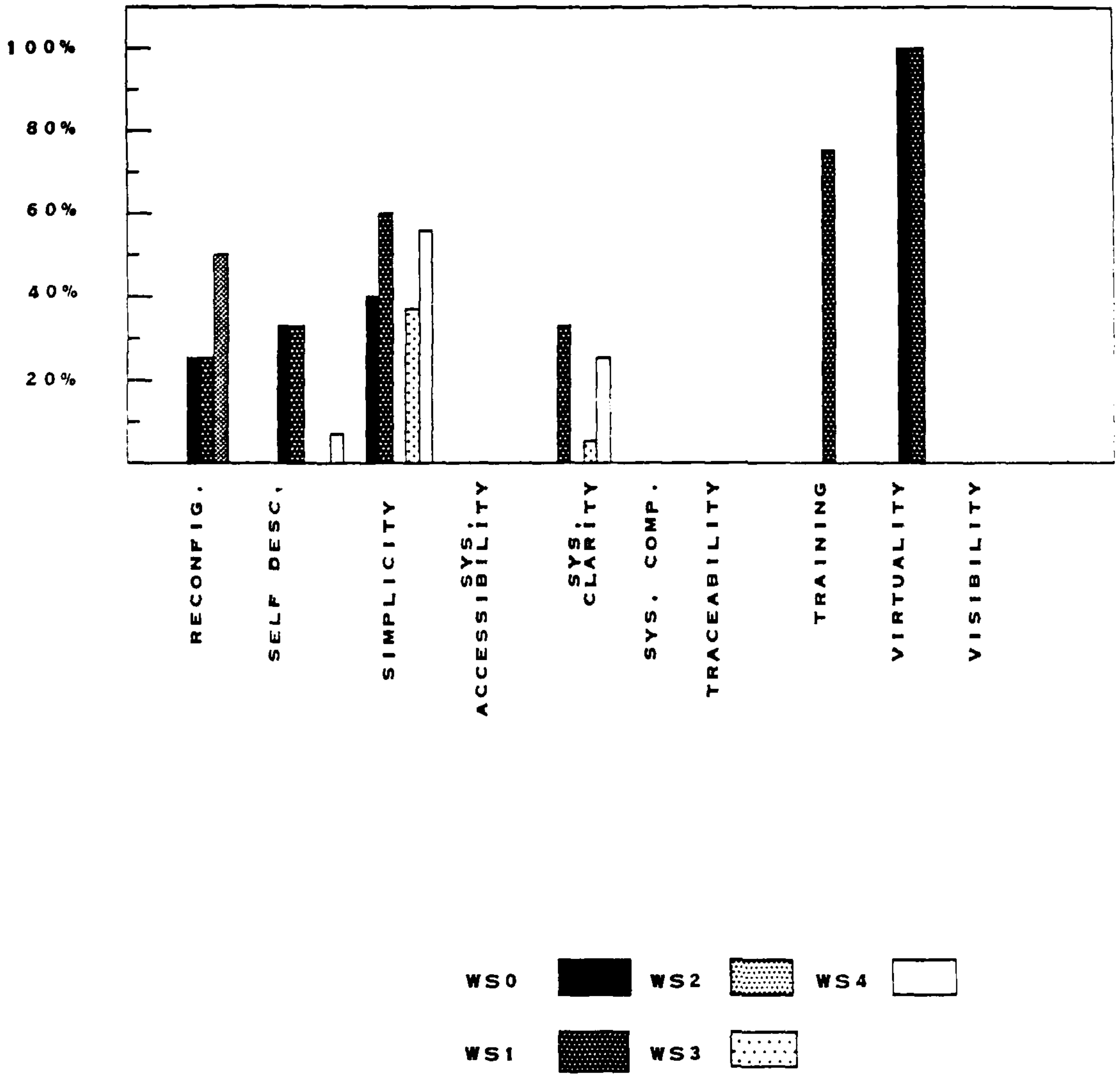


Figure 3.4-5 (Continued)

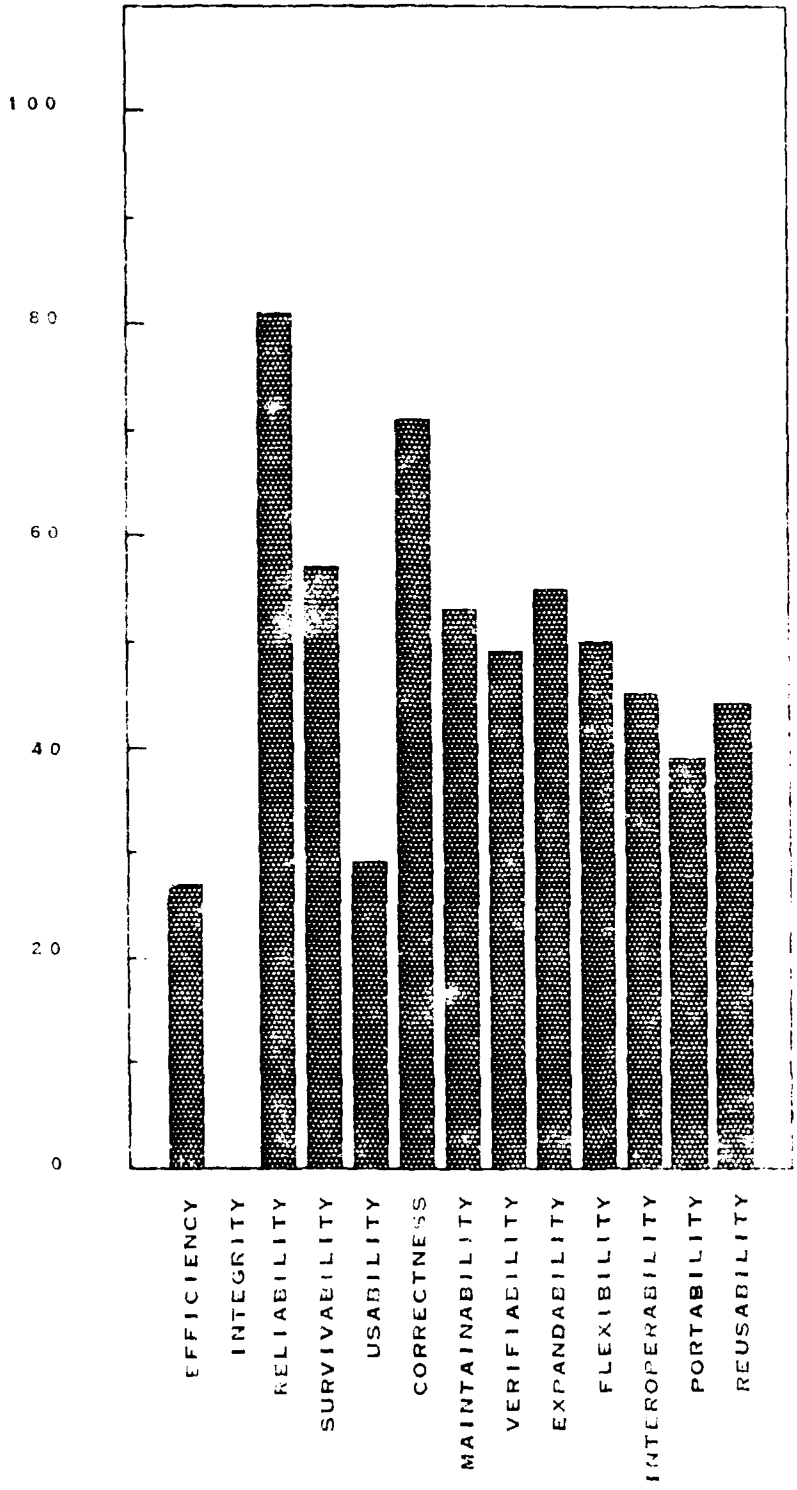


Figure 3.4-6 Percentage of Quality Factor Elements Which Generated Comments

highest percentage of metric elements reported was 80% for the factor RELIABILITY. This was partly due to the high number of questions reported for the criterion anomaly management. Figure 3.4-6 displays this data for all of the factors.

3.5 Metric Scoring

Based on the results of the worksheet application for the decision aids and phases, scores were calculated for each aid for the requirements, preliminary design, detailed design, and coding phases. Scores were calculated in two fashions: one method counted every question, every metric element, every criteria, and every quality factor. The other method counted elements as specified in the SQM methodology. Both of these methods were used because of the research nature of this evaluation project. All questions were counted in order to gather data about every metric element, and the specified questions were counted in order to follow the guidebook methodology as closely as possible.

Section 2.0 contains the results of this scoring process. As was expected (because of the quality of the available documentation), scores did not achieve their desired levels. A discussion of this problem is included in Section 2.

During the scoring process, only one new problem was identified. During use of the scoresheet for the EFFICIENCY factor, an error in the format was discovered. This is a presentation error only, and can easily be corrected. The recommended correction appears in Section 4.4.4.

4.0 RECOMMENDATIONS AND CONCLUSIONS

Based on problems encountered during software quality evaluation, SAIC has compiled a set of conclusions and recommended methodology modifications. These conclusions and modifications are discussed in this section, which is organized as follows:

- Section 4.1 contains conclusions drawn by SAIC based on the project work performed.

- Section 4.2 presents general recommendations for changes or additions to the methodology which are applicable to the entire acquisition process.

- Section 4.3 includes recommendations for modifications to the methodology which are applicable to the specification of software quality requirements. These correspond to changes recommended for specific paragraphs in Volume II of the guidebooks.

- Section 4.4 discusses recommendations applicable to software quality evaluation. These correspond to changes recommended for specific paragraphs in Volume III of the guidebooks.

4.1 Conclusions

SAIC believes the SQM methodology recommended in the guidebooks is sound and that it can be very useful to the acquisition manager. The manner of presentation in both volumes of the guidebooks can be improved, and specific recommendations for improvements are listed below.

The resulting estimate of system quality determined using the SQM methodology corresponds to that subjectively and intuitively determined by the analysts assessing both aids. Analysts felt that the documentation and structuring of the code were such that the low numerical scores received were justified. However, these scores do not mean that both decision aids are "bad." The aids were intended to be prototype, proof-of-concept developments. They were never meant to be used in the field, nor to access actual intelligence data bases. The aids were intended only to show the usefulness and meaningfulness of the concepts in battle staff management. For this reason, it is not likely that the added expense of software with high-quality development was needed or appropriate.

The reaction of the software developers at Par Technology is also indicative of barriers to be crossed before the SQM methodology can be accepted across the DoD and industry. When presented with many of the metric violations, the developers often seemed to feel that there was no need for the system to meet such a standard because "of course it would be included." For example, commenting need not be required because any good developer heavily comments code. When measured, however, we often found that they did not in fact follow this basic standard. Another example is in the length of modules -- some were over 1000 lines long and only lightly commented. We believe that pressures and schedules forced short-cuts that were unnoticed because only implicit standards existed. When the development and acquisition communities accept explicit standards and assessment, this problem will be greatly reduced.

4.2 General Recommendations

This paragraph contains recommendations that are general in nature, and apply to the methodology spanning the entire acquisition process. They do not reflect any particular paragraph or area in the SQM guidebooks.

4.2.1 Guidebook Reorganization

The SQM guidebooks are written in a style that mixes theory with the methodological steps to be performed. There is some separation in the organization of both guidebooks, with Section 4.0 of each basically containing the procedures to be followed for quality specification and evaluation. However, it would be better to further separate these elements. Section 4.0 of both volumes should be directly concerned with procedures to be executed, and contain clearly labelled examples of each of the steps. The theory and justification for the steps, along with explanations of the measurement technology, should all be contained in the earlier sections of each volume. We recommend retaining the "stand-alone" nature of each volume, with both volumes containing theory and explanation as needed.

4.2.2 Continued Research

We also recommend that experimental work be continued to provide a "real-world" basis for the contentions of the measurement methodology. The techniques and procedures described will be much more acceptable to acquisition managers and developers when experimentally verified as being cost-effective and beneficial to projects.

It is very important that work be continued to establish the relationship between the SQM methodology measurements of quality and the perceived, real-world quality of systems. We need to validate what scores mean and how important relative differences are. We need to provide a method for the acquisition manager to understand what he is receiving when a score for quality is measured at .78, for example, as opposed to .95. This double focus on specifying what the manager requires, and providing information as to what that means in real-world terms, is important. No project exists in isolation, and methods for providing measurements and information across projects is important.

Until we have full data for determining what scores and results mean in absolute terms, we are concerned the the SQM technology will appear arbitrary. Metric question weights and contribution to total scores is one area that reflects this concern. For example, consider the computation of the criterion anomaly management, used to determine the factor RELIABILITY. At the preliminary design phase (Worksheet 2), the Communication Errors metric consists of four metric elements (AM.6(1) through AM.6(4)). At both the system and software requirements analysis level and at the CSCI requirement analysis (Worksheet 0 and Worksheet 1, the only metric element that comprises the Communication Errors metric is AM.6(1). This means that at the first level of scoring the value of AM.6(1) is automatically weighted at preliminary design as 1/4 of that of the requirements analysis phase, merely because there are 3 additional metric elements for the preliminary design phase.

Another area of research concerns the metric questions that are applicable to such areas as decision aid and knowledge-based systems, as well as such projects as data base management systems. These more specialized systems have structures that are not easily measured under the more Fortran-oriented existing metric element questions. Examples of these systems are those containing data base management languages, rule-based systems, systems using non-von Neumann architectures, highly distributed systems, and systems written using non-procedural languages.

Some of the metric factors, such as INTEGRITY, do not have questions across all phases. These voids indicate that we need to look at specific areas in order to generate more metric elements and metric element questions. Each factor should be traceable through each phase of the development process, and should have representing questions on each level of worksheet. Further studies, particularly of the newer systems described above, will help fill this gap.

Automation of metric evaluation is important, and is an area already undergoing research and development. In addition to analyzing the types of questions that may be counted and evaluated automatically, we believe consideration should be given to the questions that provide the most information (see Section 4.3.4) and the skill level of those analysts who can answer the questions (see Section 4.4.1). In addition, we recommend that efforts concentrate on the questions contained in Worksheets 3B and 4B. This is because of the application of these worksheets. Questions on earlier worksheets are completed once for the system, or once for each CSCI. Even in large systems, the number of CSCIs is generally small, and question evaluation relatively rapid. For even smaller projects, however, the unit-level questions of 3B and 4B are repeated a great many times. Each of these questions must be answered for every single unit in the system, and this can become an extremely time-consuming tasks. Any automation efforts on those worksheets will provide a high level of return in reduction of human labor requirements.

A great deal of work has been completed lately concerning the guidebook methodology and other metric efforts. This work should be integrated into a revised and enhanced approach and guidebook.

4.2.3 Training

SAIC recommends that seminars be developed and offered to acquisition managers to train them in the use of the software quality measurement methodology. A one or two-day seminar would allow the acquisition manager to receive materials on the methodology, to be able to discuss its use, to ask questions directly of experts, and to engage in round-table discussions about the methodology and its worth and usefulness. Such seminars or classes make a great difference in understanding a system and its use. It is easier to understand even technical and detailed instructions once a thorough background has been acquired. This will also help motivate the acquisition manager to use the methodology and continue his education about its featuies.

The specifying of software quality goals would be aided by automating the goal specification process in the form of an expert or knowledge-based system. This automation would help the acquisition manager effectively and quickly specify quality goals. A class and demonstration for this system would be effective in increasing acceptance of the metric measurement process.

In addition, we also recommend training classes and/or materials be created for those who will be collecting data and evaluating the worksheets. This training would greatly aid in reducing subjectivity, and in allowing evaluators to share their knowledge and experiences. The automation of metrics, using such tools as the Automated Measurement System, will also be enhanced by providing training. The training could increase SQM technology effectiveness, and thereby increase its acceptance by the software community in general.

4.2.4 Framework Modifications

SAIC recommends that the basic metric framework be retained, but does have some specific recommendations to improve it. The recommended changes are minor, but we do not advocate more substantial changes until more data has been gathered validating the framework. This data will be used to support the relationships currently existing (metric to criteria to factor), and will aid in establishing new relationships.

One recommended area concerns the quality factors EFFICIENCY and INTEROPERABILITY. As currently implemented, both of these factors basically measure how well the system meets its own requirements (e.g., is as efficient as it was required to be, or operates with other system as required). Neither factor provides information in absolute terms. INTEROPERABILITY, in particular, does not assess the future ease of connecting to other systems, but measures only if the system can be connected to those specified in its own requirement documentation. SAIC recommends that more research be conducted concerning these two factors in particular. It may be that both should be removed from the factor level of the framework, and placed under the factor CORRECTNESS as criteria.

In addition, two of the quality factors have few metric element questions to calculate their values. Both INTEGRITY and USABILITY need to be evaluated for further criteria and metric question creation.

4.3 Quality Goal Specification Recommendations

Recommendations and conclusions contained in this section are applicable to Volume II of the SQM guidebooks, the <u>Software Quality Specification Guidebook</u> [BOE-2].

4.3.1 New Factor Relationships

See: Guidebook Volume II, Para. 4.1.3.2, "Quantification of Relationships"

One of the early tasks to be performed by the acquisition manager is the specification of goal scores for the quality factors. Part of this process is the consideration of the relationship among quality factors. This relationship can be of a positive nature, or it can be negative.

The software quality measurement (SQM) methodology states that the factor EFFICIENCY is interrelated with all other factors (except CORRECTNESS) in a particularly negative way. This means that the more efficient a system is, the less high its quality can be when measured for other factors. Volume II of the guidebook presents information in Table 4.1.3-3 that shows these negative interrelationships. EFFICIENCY has varying negative impacts on every other quality factor, except for the factor CORRECTNESS. These contentions seem to be intuitively valid.

The measurement technology (i.e., worksheet evaluation), however, does not support this in any way. The only criteria applicable to the factor efficiency are effectiveness-communication, effectiveness-processing, and effectiveness-storage. These relate to no other quality factor. The metric elements related to these criteria are not used for any other criteria.

The nature of these metric element questions, summarized for reference in Table 4.3-1, does not bear out the contention of negative impact. The complete satisfaction of each question (i.e., a score of "1" or "yes" for all elements) would not cause a negative impact on any other factor.

A high quality score on EFFICIENCY and simultaneously on other quality factors is perfectly possible given the present framework. We believe, however, that EFFICIENCY does have a negative relationship to other factors. This negative relationship should be reflected in the SQM framework. (Positive relationships are discussed at the end of this section.)

Any negative relationships among the factors should be supported by the sharing of common metric elements. If EFFICIENCY has a negative impact

TABLE 4.3-1 SUMMARY OF EFFICIENCY METRIC ELEMENTS

WORKSHEET/MNEMONIC		QUESTION
0	EC.1(1)	Performance requirements for communication
	EP.1(1)	Performance requirements for processing
	EP.1(3)	Optimizing comipler or assembly lang.
	EP.1(5)	Overlays required
	EP.2(1)	Data storage and processing
	EP.2(2)	Efficient processing required
	EP.2(3)	Source code supporting variable initialization
	ES.1(1)	Data storage requirements
	ES.1(2)	Virtual Storage
	ES.1(5)	Dynamic memory management
	ES.1(7)	Optimizing compiler
	ES.1(8)	Avoid redundant storage
1	EC.1(1)	Performance requirements for communication
	EP.1(1)	Performance requirements for processing
	EP.1(3)	Optimizing compiler or assembly lang.
	EP.1(5)	Overlays required
	EP.2(1)	Data storage and processing
	EP.2(2)	Efficient processing required
	EP.2(3)	Source code supporting variable initialization
	ES.1(1)	Data storage requirements
	ES.1(2)	Virtual Storage
	ES.1(5)	Dynamic memory management
	ES.1(7)	Optimizing compiler
	ES.1(8)	Avoid redundant storage
2	EP.1(5)	Overlays used
	EP.2(2)	Storage organized for efficient processing
	EP.2(3)	Source code allow variable initialization
	EP.2(6)	Efficient processing of related similar items
	ES.1(2)	Virtual storage
	ES.1(5)	Dynamic memory management
	ES.1(8)	Free from redundant storage
3	EP.1(2)	Loops with non-loop dependent statements
	EP.1(4)	Compound expressions recalculated needlessly
	EP.1(6)	Bit/Byte packing/unpacking needlessly in loops
	EP.2(4)	Arithmetic expressions with different size items
	EP.2(5)	Mixed data types in arithmetic expressions
	EP.2(7)	Data item modified
	ES.1(6)	Data packing operations
4	EP.1(2)	Loops with non-loop dependent statements
	EP.1(3)	Unit optimized for processing efficiency
	EP.1(4)	Compound expressions recalculated needlessly
	EP.1(6)	Bit/Byte packing/unpacking needlessly in loops
	EP.2(4)	Arithmetic expressions with different size items
	EP.2(5)	Mixed data types in arithmetic expressions
	EP.2(7)	Data items modified
	ES.1(6)	Data packing operations

on RELIABILITY, then common elements should ensure that a higher score in one means a lower score in the other. Otherwise, the factors are independently measured and no relationship attributes need to be considered.

This will likely mean that questions or elements must be considered in pairs. As an example, consider this question from Worksheet 4B: MO.1(8) "Is temporary storage (i.e., workspace reserved for intermediate or partial results) used only by this unit during execution (i.e., is not stored with other units)?" For the criterion modularity, this question would contribute in a positive fashion. A "yes" answer would receive a score of 1. In contrast, it is a more efficient use of memory space to share storage. This means that a "no" answer should receive a score of 1 relating to the criterion effectiveness-storage. To accomplish this, it is possible that the question could be paired, as shown below.

> MO.1(8) Is temporary storage (i.e., workspace reserved for intermediate or partial results) used only by this unit during execution (i.e., is not stored with other units)?
>
> ES.x(n) Does this unit share temporary storage (i.e. workspace reserved for intermediate or partial results) with other units during execution?

This "paired" approach fits in effectively with the workbook methodology we recommend below (paragraph 4.4.1) for data collection. The data would be collected once using the workbook, and then used to answer each of the two separate questions. This means that no additional work would be required to gather the paired data items.

Existing questions can be used to build this interconnection among related factors. Table 4.3-2 is a partial list of these existing elements, along with the newly-developed "pair" element. These elements are too few to fully justify the relationships we believe exist. This means that more questions are needed for each factor so that calculated values will correspond to real-world observations.

One advantage of this paired relationship is in its effect on the tendency of metric methodology users to try to force every score to a value of "1" or "yes." Some users have believed that the purpose of the technology is to create a resulting system that has every metric element question answerable with a "yes" or a full score of "1." The technology actually goes beyond that, and is an attempt to reflect system quality in a cost-effective manner. This means that some "lacks" in a system *should not be corrected*, because the cost would not justify the benefits. Paired metric questions would not allow all scores to reach "1", and would better reflect the situations existing in industry today.

Also to be considered in this regard are the complementary factor relationships described by the guidebooks. Four quality factors (RELIABILITY, CORRECTNESS, MAINTAINABILITY, and VERIFIABILITY) are

TABLE 4.3-2 "EFFICIENCY" SAMPLE PROPOSED QUESTIONS

WORKSHEET/MNEMONIC	QUESTION
AM.1(2)	a. How many error conditions are required to be recognized (identified)? b. How many recognized error conditions require recovery or repair? c. Calculate b/a and enter score.
EP.x(n)	a. How many error conditions are required to be recognized (identified)? b. How many recognized error conditions require recovery or repair? c. Calculate 1-b/a and enter score.
AM.1(4)	a. How many instances of the same process (or function, subfunction) being required to execute more than once for comparison purposes? b. ... c. ...
EP.x(n)	a. How many instances of the same process (or function, subfunction) being required to execute more than once for comparison purposes? b. Calculate 1/(a+1) and enter score.
AM.3(2)	Are there requirements to range test all critical (e.g., supporting a mission-critical function) loop and multiple transfer index parameters before use?
EP.x(n)	Is the CSCI free from requirements to range test all loop and multiple transfer index parameters before use?

TABLE 4.3-2 "EFFICIENCY" SAMPLE PROPOSED QUESTIONS (Con'd)

WORKSHEET/MNEMONIC	QUESTION
AM.3(3)	Are there requirements to range test all critical (e.g., supporting a mission-critical function) subscript values before use?
EP.x(n)	Is the CSCI free of requirements to range test all critical (e.g., supporting a mission-critical function) subscript values before use?
AM.7(2)	Are there requirements to periodically check all adjacent nodes or interoperating systems for operational status?
EP.x(n)	Is the CSCI free from requirements to periodically check all adjacent nodes or interoperating systems for operational status?
AP.4(1)	Is there a requirement to avoid or to limit the use of microcode instruction statements?
EP.x(n)	Is the CSCI free from a requirement to avoid or limit the use of microcode instruction statements?

described as being complementary to all other factors. Low scores on any of these factors mean that other quality factors, even if high scores are achieved, have to be of lower quality. SAIC agrees with recommendations by other contractors that this complementary relationship should not be included in the quality factor framework.

Factor quality scores should indicate quality, regardless of the scores achieved for other factors. This does not mean that the factors have no overlap, but all measures necessary for a factor should be evaluated with that factor. Measures should not depend on other scores (e.g., INTEGRITY is not high unless the four factors listed above are also high). We recommend that the applicable criteria of the four factors that are considered to be complementary be included in each quality factor in the framework, so that each may stand alone. This common set of data will likely ensure that the score for a factor such as INTEGRITY cannot be high if scores for the four complementary factors are low. Research efforts will need to identify what these applicable criteria are.

4.3.2 Quality Factor Definition

See: Guidebook Volume II, Paragraphs 3.1.1 and 4.1.2.3, "Factor Definitions and Rating Formulas", and "Quality Requirements Survey"

The SQM guidebook defines software quality factors in two tables (Volume II, Tables 3.1-1 and 4.1.2-4). Volume III, in Tables 3.1-1 and 3.1-2, contains the same information. These definitions, shown here in Table 4.3-1 for reference, are very misleading. There is absolutely no evidence to date that the calculated scores for EFFICIENCY, for example, relate to the formula given in the methodology. This formula (the complement of the ratio of actual to allocated usage) may indeed be a valid measurement of efficiency, but in no way relates to the numbers calculated for the factor EFFICIENCY. The implication in these tables is that specifying a goal of .9 for EFFICIENCY, and receiving a score of .9 using the worksheets, means that the actual measured resource utilization is .9 that of the allocated usage. This conclusion has not been verified by data. If one includes the arbitrary weighting of criteria, as well as the tailored selection of metric elements, the relationship is even more difficult to prove.

The use of these definitions by those new to software measurement technology can cause difficulty. SAIC, as an example, as been involved in the development of the Joint Forward Air Defense Test Bed (JFAAD) for the Air Force. Using the SQM guidebooks, the JFAAD project office requested that the system be built such as to achieve a final quality rating of .996 for RELIABILITY. They calculated this rating, using guidebook Table 3.1-1, by deciding that 4 errors in each 1000 lines of code was an acceptable number. They expected that the RELIABILITY scores achieved at each phase would reflect that number, and result in a fielded system that had 4 or less errors for each 1000 lines of code. At this point in time, there is no data to support that relationship.

Table 4.3-3 Quality Factor Ratings

Quality factor	Rating formula	Rating information			
Efficiency	$1 - \frac{\text{Actual utilization}}{\text{Allocated utilization}}$	Value	0.1	0.3	0.5
		% utilization	90%	70%	50%
Integrity	$1 - \frac{\text{Errors}}{\text{Lines of code}}$	Value	0.9995	0.9997	0.9999
		Errors/LOC	5/10,000	3/10,000	1/10,000
Reliability	$1 - \frac{\text{Errors}}{\text{Lines of code}}$	Value	0.995	0.997	0.999
		Errors/LOC	5/1,000	3/1,000	1/1,000
Survivability	$1 - \frac{\text{Errors}}{\text{Lines of code}}$	Value	0.9995	0.9997	0.9999
		Errors/LOC	5/10,000	3/10,000	1/10,000
Usability	$1 - \frac{\text{Labor-days to use}}{\text{Labor-years to develop}}$	Value	0.5	0.7	0.9
		Days/years	5/10	6/20	10/100
Correctness	$1 - \frac{\text{Errors}}{\text{Lines of code}}$	Value	0.9995	0.9997	0.9999
		Errors/LOC	5/10,000	3/10,000	1/10,000
Maintainability	1- 0.1 (average labor-days to fix)	Value	0.8	0.9	0.95
		Average labor-days	2.0	1.0	0.5
Verifiability	$1 - \frac{\text{Effort to verify}}{\text{Effort to develop}}$	Value	0.4	0.5	0.6
		% effort	60%	50%	40%
Expandability	$1 - \frac{\text{Effort to expand}}{\text{Effort to develop}}$	Value	0.8	0.9	0.95
		% effort	20%	10%	5%
Flexibility	1- 0.05 (average labor-days to change)	Value	0.8	0.9	0.95
		Average labor-days	40	20	10
Interoperability	$1 - \frac{\text{Effort to couple}}{\text{Effort to develop}}$	Value	0.9	0.95	0.99
		% effort	10	5	1
Portability	$1 - \frac{\text{Effort to transport}}{\text{Effort to develop}}$	Value	0.9	0.95	0.99
		% effort	10	5	1
Reusability	$1 - \frac{\text{Effort to convert}}{\text{Effort to develop}}$	Value	0.4	0.6	0.8
		% effort	60	40	20

As research continues over time, formulas will be developed that show what quality measurement values mean in terms of such things as resource utilization, errors per line of code, effort to fix, and effort to convert. At the present time, however, it is very misleading to imply that we know what quality scores mean in terms of absolute numbers. Currently, they are relative indicators only. We recommend, therefore, that any reference to these rating formulas be removed from the guidebooks.

In replacement for these formulas, we recommend using more concrete examples and descriptions of the factors. These descriptions can be drawn from the criteria and metric levels of the measurement framework. As an example, consider the quality factor MAINTAINABILITY. If the acquisition manager is specifying that he wishes a system to be maintainable, and if he is to later accept measurement of the level of achievement of that factor, then he should have some idea of what is considered to make up a maintainable system. This can be described in terms of what is going to be used to assess how much "maintainability" is present.

The products of the development process that are assessed are both documents and software code itself. If the system is to be maintainable, it means then that the documents and the source code should have characteristics that promote maintainabililty. Using the characteristics that are evaluated has the added advantage of giving the acquisition manager more insight into what is being measured. Table 4.3-4 is an example of how this might be accomplished.

Documents should be accessible, well-structured, clearly and simply written, depict control and data flow, be indexed, be separated by system functions, and list all operational capabilities. Standards should require such things as commenting global data and commenting variables. Code should be structured, indented, of reasonable size, etc. The framework indicates that items having these characteristics are easier to maintain. A table that shows the product and the attributes that make it of higher quality would be useful. A non-software oriented person should be able to use these tables to understand what is being assessed and what the quality factors indicate, because the technical content need not be high. The data would provide a positive look at what is measured, and what the assessment numbers mean.

4.3.3 Factor/Criteria Interrelationships

See: Guidebook Volume II, Paragraph 4.1.3.1, "Shared Criteria"

The acquisition manager correlates system quality factors with software quality factors, and additionally correlates the software quality factors with the criteria which constitute them. The present methodology does not provide adequate definitions of these software quality factors in terms of the criteria. The acquisition manager who

TABLE 4.3-4 SAMPLE "MAINTAINABILITY" DESCRIPTION

PRODUCT	CHARACTERISTICS ENHANCING MAINTAINABILITY
All Documentation	Accessible
	Wel structured
	Clearly and simply written
	Indexing scheme used
Software Standards and Procedures Manual	Require code to have:
	commented global data
	comment variables
	Design standards for unit prologues, comments, *and unit structures*
Software Requirements Specification	Depict control and data flow
	List all software operational capabilities
Software Top Level Design Document	Standardized design representation
	Calling sequence protocol established
	External I/O protocol and format established
	Error handling required
	Functions always referenced by same name
	Data representation standardized
	Data naming consistent
	Global data defined
	Consistent calling sequence parameters
	Use structured design techniques
	Comply with standards
Software Detailed Design Document	Follow standards
	Units each have single name
	Unit size small
	Control variable passing minimized
	Local storage
	Single objective in each unit
	Single entrance and exit in each unit
	Branching levels low
	Control flow is top to bottom
	Few negative or compound Boolean expressions
	Few loops with unnatural exits
	No self-modifying units
Source Code	Comply with standards
	Unique data names
	Commented
	Written in a high order languare
	Data names descriptive
	Indented and blocked logically
	Single exit and entrance in each unit

attempts to thoroughly understand the impact of his direction at the factor level upon the criteria at the design level must understand definitions that are likely to be too technical.

Ideally, goal specification could be conducted using an automated decision aid which employs optimization techniques. Short of this enhancement to the present methodology, providing a graphic representation of the interrelationships would enable goal specification to be more effective. The interrelationships included would be those between each quality factor (and component criteria) and the other influencing quality factors and criteria.

Figure 4.3-1 combines the information contained in several SQM guidebook figures and tables, and represents the quality factor FLEXIBILITY, as an example. One such figure for each quality factor would provide an improvement over the complex methodology currently used. The figure is broken into four quadrants: the two left quadrants are associated with factors, the two right quadrants are associated with criteria, the two upper quadrants are associated with positive influence, and the two lower with negative. Arrows into the subject quality factor and its criteria indicate influences upon them; arrows out indicate the subject factor and its criteria's influence upon other factors and criteria. This figure is shown as an example, and it is possible to include other pertinent information. As an example, costs associated with factors could be included in a graphic representaion.

Using this figure alone, it is easy to comprehend that the quality factor FLEXIBILITY is positively influenced by the quality factors of CORRECTNESS and MAINTAINABILITY, and negatively influenced by SURVIVABILITY. Additionally, the figure indicates the criteria that determine FLEXIBILITY and those others which influence FLEXIBILITY. The criteria of consistency and traceability (in the "positive criteria" or upper right quadrant) have a positive influence on FLEXIBILITY, while reconfigurability (in the "negative criteria" or lower right quadrant), has a negative influence. This figure demonstrates nine types of relationships:

- Positive factors influencing this factor
- Negative factors influencing this factor
- Positive criteria influencing this factor
- Negative criteria influencing this factor
- Other factors which are positively influenced by this factor
- Other factors which are negatively influenced by this factor
- Other criteria which are positively influenced by this factor

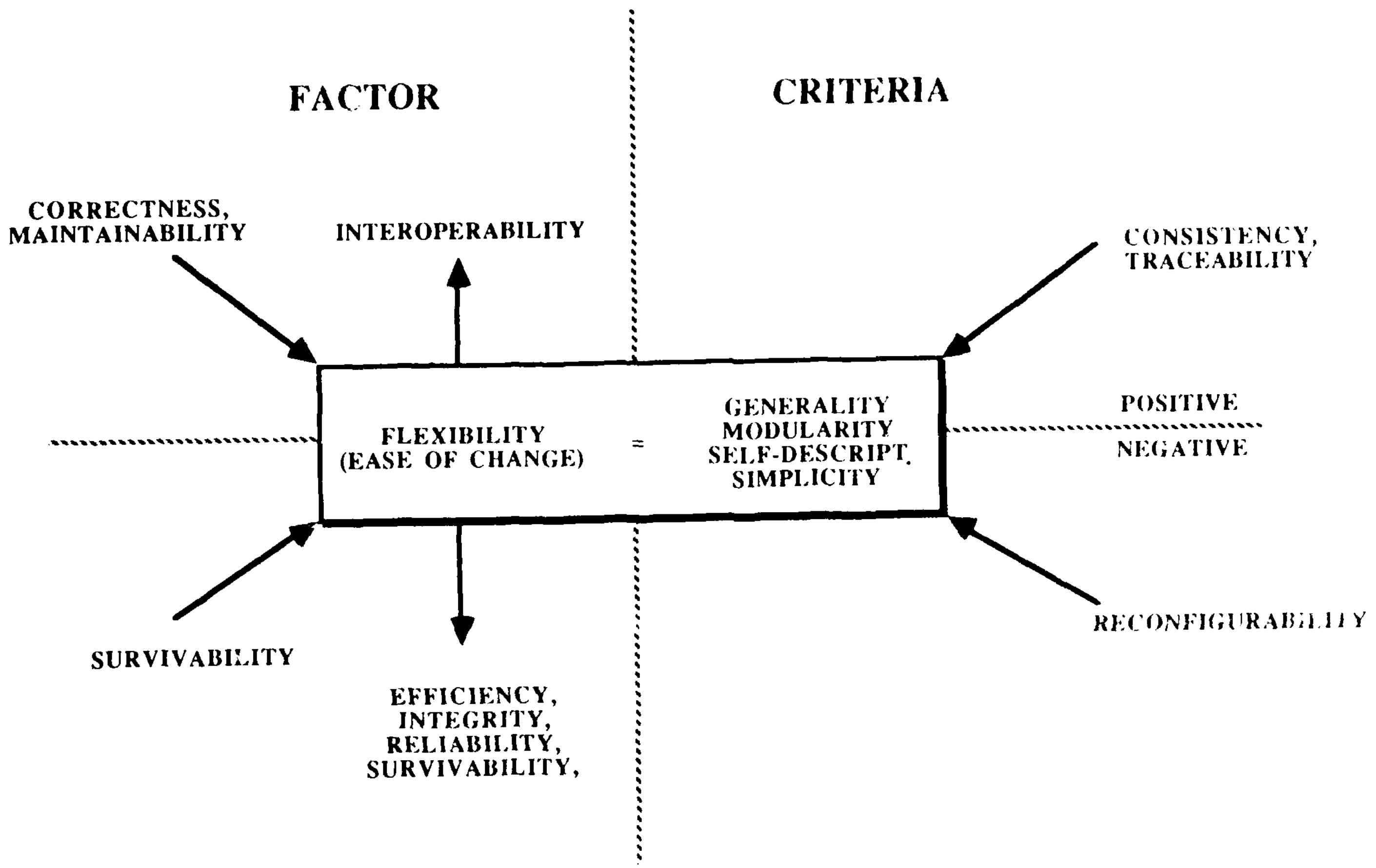


FIGURE 4.3-1 FLEXIBILITY QUALITY FACTOR/CRITERIA INTERACTION DIAGRAM

- Other criteria which are negatively influenced by this factor

- Criteria which make up this quality factor

Cost ranges could also be included in the figure to indicate the range of cost for the subject quality factor for system acquisition phases, and additionally indicate the cost impact for each phase for each quality factor that influences the subject factor. This would enable the individual tasked with goal specification to see not only the cost impact of the emphasis of the subject quality factor, but also the cost impact of each of the other factors influencing the subject factor positively or negatively, for each acquisition phase. These ranges could be presented graphically, for ease of use.

4.3.4 Minimum Effective Set of Measurements

See: Guidebook Volume II, Paragraph 4.1.4, "Consider Costs"

Another method to aid the acquisition manager in the specification of software quality goals would be to aid him in determining the most cost-effective minimum subset of factors, criteria, and metrics to be used.

Figure 4.3-2 displays a matrix of the quality criteria and the quality factors to which they apply. There are 29 criteria used in determining 13 quality factor scores. The distribution of the number of quality factors per criteria is displayed in the figure. There are 53 intersections of factors and criteria represented, which we call "cells" for the purpose of this analysis. Those criteria which are employed in the determination of 3 or more quality factors actually account for 49% of the 53 cells in the matrix. Thus, temporarily ignoring the number of questions per cell and the associated relative difficulty in providing the answers, the questions relating to the criteria of generality, independence, modularity, self-descriptiveness, and simplicity provide almost one half of the quality factor information. These criteria are only one sixth of those measured. Similar analysis reveals that modularity alone provides 15% (8 cells of 53) of the information, and modularity, self-descriptiveness, and simplicity together provide 38% (20 cells of 53) of the information indicated by "x's." These relationships are indicated in Figure 4.3-3.

Each criteria is scored by averaging the values assigned to its applicable metric element questions. The actual number of metric element questions asked for each criteria is not significant. Because of the selection process, varying numbers of questions will be scored across projects as the methodology is used. For this reason, we are not analyzing here the number of questions used to compute each of these multi-factor criteria.

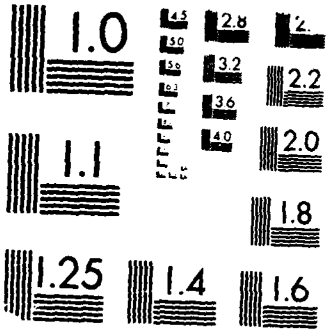
For reference, however, there are 15 self-descriptiveness metric elements, 30 simplicity metric elements, 5 generality metric elements, CRITERIA

FACTORS	ACCURACY	ANOMALY MGMT.	APPL. INDEPENDENCE	AUGMENTABILITY	AUTONOMY	COMMONALITY	COMPLETENESS	CONSISTENCY	DISTRIBUTEDNESS	DOCUMNT. ACCESS.	EFFECT. COMMUN.	EFFECT. PROC.	EFFECT. STORAGE	FUNCT. OVERLAP	FUNCT. SCOPE	GENERALITY	INDEPENDENCE	MODULARITY	OPERABILITY	RECONFIGURABILITY	SELF DESCRIPT.	SIMPLICITY	SYSTEM ACCESS.	SYSTEM CLARITY	SYSTEM COMPAT.	TRACEABILITY	TRAINING	VIRTUALITY	VISIBILITY
EXPANDABILITY				●												●		●			●	●						●	
INTEROPERABILITY						●								●			●	●							●				
REUSABILITY			●							●					●	●	●	●			●	●		●					
PORTABILITY																	●	●			●								
FLEXIBILITY																●		●			●	●							
VERIFIABILITY																		●			●	●							●
SURVIVABILITY		●			●				●									●		●									
USABILITY																			●								●		
INTEGRITY																							●						
EFFICIENCY											●	●	●																
MAINTAINABILITY																		●			●	●							●
RELIABILITY	●	●						●														●							
CORRECTNESS							●	●																		●			

CRITERIA CONTRIBUTION TO QUALITY FACTORS

FIGURE 4.3-2 SOFTWARE QUALITY - FACTOR vs. CRITERIA

AD-A192 849 SOFTWARE QUALITY MEASUREMENT DEMONSTRATION PROJECT II 2/2
(U) SCIENCE APPLICATIONS INTERNATIONAL CORP SAN DIEGO
CA P PIERCE ET AL. OCT 87 RADC-TR-87-164
UNCLASSIFIED F30602-81-C-0132 F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART

BUREAU OF STANDARDS 1963 A

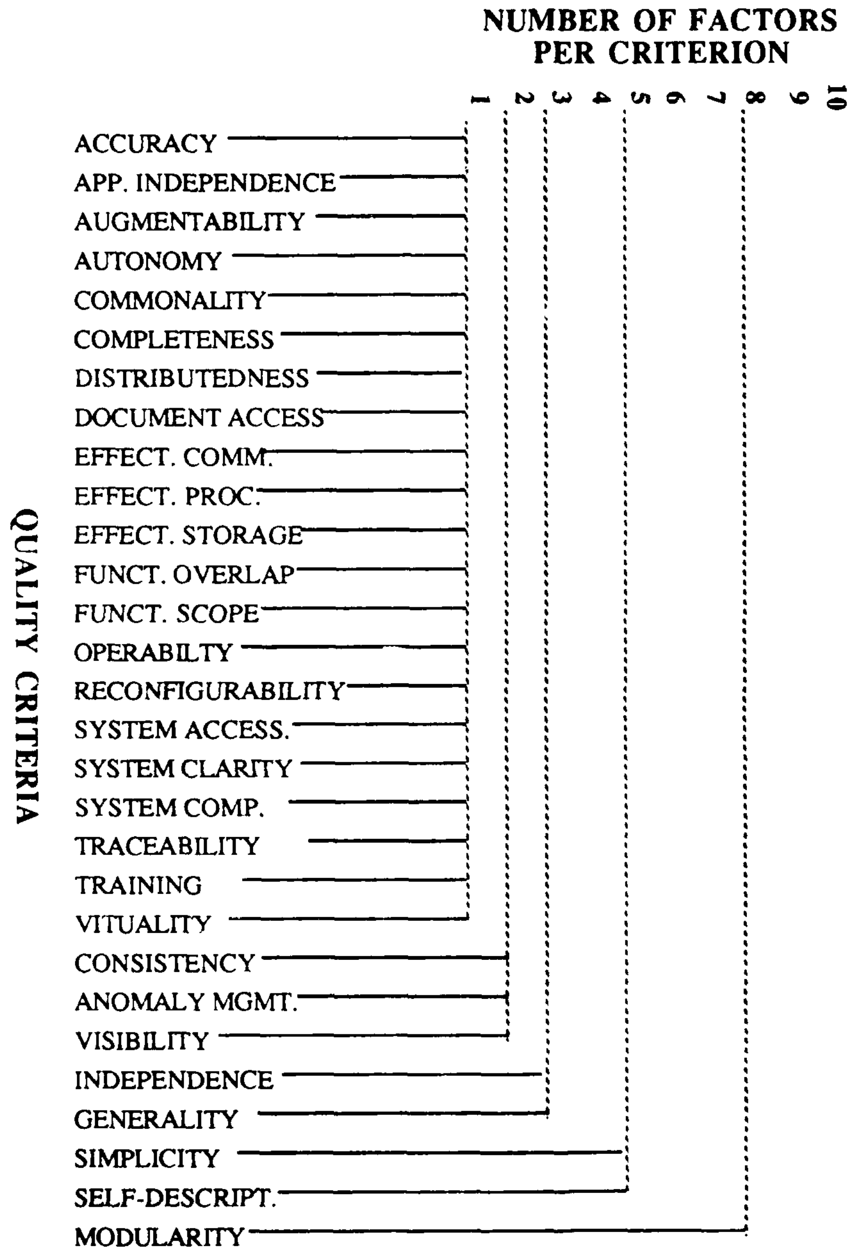


FIGURE 4.3-3 CRITERION VERSUS NUMBER OF FACTORS PER CRITERION

14 modularity metric elements, and 7 independence metric elements. There are a total of 327 metric elements over the worksheet set. This means that 5 of 29 criteria (17%) contain 71 of 327 metric elements (21%), and make up 49% of the factor scores when criteria weighting is not included. One implication of this is that these 49% of the metric elements might be prime candidates for automating data evaluation. While some metrics are relatively easy to automate, even the more sophisticated metrics would provide a great deal of information if selected from this set.

A potentially viable alternative to full-scale software quality evaluation with seven worksheets would be to evaluate a reduced set of quality criteria. Since five criteria (generality, independence, modularity, simplicity and self-descriptiveness) include almost half of the total number of metric elements, they alone could be used. This would provide no information for the factors CORRECTNESS, EFFICIENCY, INTEGRITY, and USABILITY, but partial information would be provided for all the other factors.

Another approach that could be suggested to acquisition managers is to measure, in the early stages of development (perhaps up to and including preliminary design), only those criteria whose quality factors have been emphasized as requiring excellence. This could be used to drive detailed design, which would in turn drive coding, in the intended direction. In this manner, the evaluation required is less exhaustive and focused early to ensure that quality is high where it is most required.

The third potential approach, used already in some projects, is to select a subset of quality factors. These factors are measured fully for each development phase.

4.3.5 Functional Allocation

See: Guidebook Volume II, Paragraphs 4.1.1 and 4.4.1, "Identify Functions", and "Review Requirements Allocations and Evaluation Formulas"

The first step that the acquisition manager must complete is the identification of the functions of his system. To perform this process, he identifies each system function which is supported by software and which will have separate quality requirements.

The SQM methodology discusses the different quality goals possible among different system functions, and includes a sample of the functions associated with an example command and control system. In addition to these considerations, the acquisition manager must examine functions unique to software (for example: man-machine interfaces, executives, mission training, and integrated test functions).

It is up to the quality assessment evaluator to allocate these functions among the actual software Computer Software Configuration Items (CSCIs) that are developed. This process is described in paragraph 4.1 of Volume III of the guidebooks. The acquisition manager then assesses this allocation as described in Volume II, paragraph 4.4.1.

SAIC recommends that much more guidance be supplied to the manager concerning both the derivation of functions for which quality goals are set and for assessing the requirement allocations and evaluation formulas.

One possible approach is to use a system-wide specification of quality goals unless certain criteria are met that force specifying functional goals. These criteria could include the following:

- Functional areas already have been identified. Many large systems are functionally decomposed in the original procurement documents (such as required operational concepts or purchase descriptions). Even if system development does not necessarily follow these functional dividing lines when CSCIs are created, they are a common point of reference that would allow a meaningful evaluation.

- The system is so diversified that wide ranges of quality goals are reasonable for various system functions. A large system may have such disparate needs that system-wide goals are not reasonable.

- The criticality (for example, risk to human life) is such that some functions must have high goals, but cost factors dictate that non-critical functions do not have the same goals.

The reason behind providing this guidance is that an arbitrary functional decomposition, followed by a somewhat arbitrary requirement allocation, only exacerbates the problems of determining what quality goals and scores actually mean for real world referents.

The SQM guidebook should clearly state that setting system-wide goals is reasonable and acceptable. In cases where particular goals are desired for particular functions, it may be effective to isolate those functions as CSCIs and set goals directly related to the CSCI itself.

In addition, we recommend de-emphasizing the process of allocation functional requirements among the various CSCIs of the system. We recommend using this system only when functions have been derived as described above, and then in a slightly different manner than is described in the guidebooks.

Rather than determining the percentage of each CSCI that contributes to each function, we recommend listing (for each function) all CSCIs that implement it. The scores calculated for each CSCI would then be averaged to calculate the functional score, without taking into consideration how much of the CSCI actually is performing a given function. This will reduce some of the arbitrary nature of the allocation process, in order to increase the comparability of results across projects. This will also help build information so that we may define, in terms of actual systems built and used, what are calculated quality numbers mean (e.g., errors per delivered lines of code, time to upgrade, time to transport).

4.3.6 Violation Procedures

See: Guidebook Volume II, Paragraph 4.4.4, "Review Metric Scores"

The goal specification guidebook should contain information on how the acquisition manager is to handle non-compliance with metric element questions, and with criteria and factor scores which fall below his goals. Along with the specification of the quality itself, the manager needs to outline what will be done when those specifications are not met. To do this, he needs guidance from the methodology about what procedures are best. It may be, for example, that a software problem report is to be generated for every "0" or "no" answer to a metric question on worksheets 0, 1, 2, 3A, and 4A. For some questions on 3A and 4A, it may be that a threshold figure should be specified against which violations must be written up as problem reports. Some projects have demanded a problem report for every violation in an effort to try to force the quality scores to a value of 1. This is probably not desirable, but guidance should be supplied. If the goal is to assess -- but not repair -- violations, this should be specified by the acquisition manager.

Independent verification and validation contractors could be of assistance to the acquisition manager in this regard. The experience and knowledge of these analysts could be used to help determine what is critical to the development and help to define procedures to ensure that violations are handled properly. The acquisition manager could use this information to establish the needed procedures.

4.3.7 Scaling of Scores

See: Guidebook Volume II, Paragraph 4.4.2, "Review Factor Scores"

The acquisition manager reviews quality scores at review points throughout the development process. He examines the scores, using a range of .9 to 1 as Excellent, of .8 to .89 as Good, and of .7 to .79 as Average.

Lacking experimental evidence as to what the measured scores really reflect (as yet they are only relative), we believe that this relationship and numerical range may be too limited. For some metric questions (those that are measured, for example, as 1/d), each change in the value can produce a wide variation in result (for example, from 1 to

.5 to .33). This variation reduces as the number grows large, but by then the score is well below even the .7 to .79 "average" range. We do not yet really know if there is a meaningful difference between a score of .8 and .9, nor do we know what the "unacceptable" cut-off level should really be.

From analyzing the goals set by the decision aid developers themselves, we also believe there is a tendency to set quality goals higher than they really need to be. The developers indicated several factors should have "excellent" values (which were not achieved), which we believe to be above those really needed for such a prototype system.

The result of this is that we believe that more information needs to be supplied to the acquisition manager to allow him to effectively evaluate the achieved scores, their ranges, and what those ranges actually mean.

4.3.8 Criteria Weighting

See: Guidebook Volume II, Paragraph 4.2.2, "Assign Weighting Formulas"

The process of weighting criteria is described in the SQM guidebook during the quality goal specification process. Each criteria making up the selected quality factors is evaluated, and weightings assigned to each (determining its percentage of the total factor score).

This process is very arbitrary and subjective. An acquisition manager has no theoretical justification for weighting any criteria, because we do not yet have all the information needed. We cannot currently point to a real-world meaning for the calculated values we create, but we do know that that is the direction we would like to move. We would like to be able to say that a system with a MAINTAINABILITY score of .98, for example, is going to be relatively inexpensive to maintain (especially compared to a system with a score of .45). If acquisition managers arbitrarily assign weights to the criteria that make up maintainability, then we have no way to compare scores.

If these scores are not meant to be used for comparison purposes, then why make a calculation at all? It would be logical, in that case, to use the metric worksheets as checklists only. Creating a "score" would not provide any further information for the developer or the acquisition manager. It would allow no relative assessment of a systems quality.

Until we have collected the data to validate the numbers we are calculating (i.e., what does a .93 score for CORRECTNESS mean?), SAIC recommends that the weighting process be dropped altogether. Rather than assigning arbitrary weights, we recommend that the acquisition manager be instructed to calculate factor scores by averaging the applicable criteria scores. This approach is already used, though not commented upon, in calculating the criteria and metric scores themselves. We do not currently weight metrics, metric elements, or metric element questions when they are grouped and scored. As data validates the software quality measurement methodology results, we believe that we will be able to derive weightings that should be used at all these levels, including criteria. These weightings will allow us to make objective assessments of quality that apply in the same fashion to most application systems.

In the place currently occupied by weighting criteria, we recommend supplying procedures that substitute for the weighting concept. Any non-applicable or non-desired criteria would be dropped from measurement and their respective metric elements left unscored. An example of this would be in eliminating a criterion like virtuality, used chiefly in network applications.

For factor criteria that are of unequal importance to the acquisition manager, procedures could be included to allow violations (i.e., low criteria scores) to be handled in a different fashion for those criteria. If effectiveness-processing (EP) is far more important to the manager than is effectiveness-communication (EC), then the results of scoring each of those criteria could be handled differently. A low score achieved concerning communication would be specified as acceptable, while a low score for the processing measure would require corrective measures. In this way, the important criteria are still regarded as vital, but do not contribute to making factor scores incomparable across systems.

4.3.9 Survey Questionnaires

See: Guidebook Volume II, Paragraph 4.1.2.3, "Quality Requirements Survey"

Quality survey questionnaires are sent out by the acquisition manager to collect information for quality score specification. When responding, each person surveyed lists the system functions, and then inserts the quality scores he would set for each function.

SAIC recommends that the functions, if they are to be used at all, be specified by the acquisition manager and included on the surveys. In addition, the forms in the guidebook, shown as examples, do not have space set aside for the respondent to indicate factor quality scores. We recommend adding specific room for these answers.

4.4 Quality Assessment Recommendations

Recommendations in this section apply to Volume III, Software Quality Evaluation Guidebook [BOE-3].

4.4.1 Data Collection Workbook

See: Guidebook Volume III, Paragraph 4.2.3, "Answer Worksheet Questions"

Volume III of the guidebooks, in paragraph 4.2.3, discusses how the worksheets are to be filled out for metric evaluation. The section is approximately one and one-half pages long, and briefly describes the layout of the worksheets, who should answer the questions, how to identify source material, and what to do about all-inclusive questions. It mentions that results should be reproducible, and that any judgments made and metric violations noted should be documented.

There are no detailed explanations or recommendations concerning exactly how the worksheet questions are to be answered. We recommend that such details be included in order to promote a uniform approach to software measurement technology. This uniformity will allow results to be gathered across various projects that are comparable. It will reduce the perception in industry and the academic community that this technology is arbitrary, and allow research to proceed that can correlate quality measurements to actual system performance. In addition, such details would supply guidance to allow the technology to be most effectively and efficiently used.

The text should reflect the fact that it is not the best approach to simply sit down with a document (or source code), get the worksheet, and answer each question in the order presented. The worksheets are not really worksheets, but are simply a list of applicable questions presented in order by mnemonic. This listing is important and useful, but data should not be collected in the same manner as the list is presented. This is because the worksheet questions are repetitive and interrelated. The questions are valid, but data can be gathered much more efficiently and effectively if another approach is taken.

The SQM guidebook should recommend an approach that is useful and cost-effective, minimizing manual effort and maximizing the data gathered. Automated tools are an obvious direction to take, and the methodology does not indicate how to take advantage of these tools.

One way to aid the user, whether he has access to tools or only to manual data collection, is to categorize the questions. Various types of categories would be useful to the question-answering process. Potential category methods are discussed below:

- Data necessary to answer the question
- Skill level required to answer the question
- Techniques for answering the question

Data necessary to answer the question. The SQM methodology was designed to be in accord with the software life cycle as described DOD-STD-2167. This life cycle has defined products and reviews that are recommended for each phase of development and deployment. The Data Item Descriptions (DIDs) that correspond to these products mandate certain information for each of the products created. One categorization method would be to list the correspondence between question source material (the location of where the answer should be in a standard DOD-STD-2167 type development) and the questions. An example of this is question SI.1(9) in Worksheet 0: Are there requirements for a programming standard? The Software Development Plan (DI-MCCR-80030), in a development following DOD-STD-2167, should contain this information in paragraph 5.1.6. Knowing where this information should be located would greatly aid in either automated or manual data collection.

A second aspect of knowing the source of data necessary to answer each question involves looking at the minimum amount of data required to answer all questions. As mentioned before, the worksheet questions are not totally independent of each other. Some questions are word-for-word repetitions of other questions, and some questions use data that is also used to answer other questions. This is not a defect, but reflects the reality that the criteria and factors we measure are not (and should not be) orthogonal to each other.

It does mean, however, that answering each worksheet question in order may involve doing the same work over and over. To avoid this, it would be easy to create a list of the minimum data set required to answer all questions for each worksheet.

Skill level required to answer the question. The questions on the various worksheets are not all equivalent in terms of the background and experience necessary for an analyst to quickly and effectively answer them. The most senior personnel could, of course, answer all questions. However, some questions are so straight forward as to lend themselves easily to either automation or to evaluation by much less experienced analysts. It is more cost-effective to use these techniques where possible, and to focus more senior effort on the areas where it is important. Table 4.4-1 presents our analysis of the skill level required for the various questions on Worksheet 4B. The category "junior analyst" refers to people with 1 to 4 years experience in such things as documentation, requirements analysis, and configuration management. The "senior analyst" has 5 to 15 years experience in the same sort of tasks. "Junior programmers" are design and implementation personnel, typically with 1 to 5 years experience. The "senior programmers" are highly experienced in system requirements, design, coding, and testing.

The skill level analysis was required for our evaluation of Worksheet 4B only. Earlier worksheets were such that non-programmers could easily evaluate nearly all of the questions. Examination and understanding of the code, however, required personnel experienced in programming. Based

TABLE 4.4-1

SKILL LEVEL REQUIRED TO ANSWER QUESTIONS ON WORKSHEET 4B

QUESTION	ANALYST			
	JUNIOR ANALYST	SENIOR ANALYST	JUNIOR PROGRAMMER	SENIOR PROGRAMMER
AM.1(3)			X	
AM.2(7)			X	
AP.1(1)			X	
AP.2(1)	X			
AP.2(2)	X			
AP.2(3)				X
AP.2(4)	X			
AP.3(2)	X			
AP.4(1)	X			
AT.1(1)			X	
AT.2(1)				X
AT.2(2)				X
CP.1(2)	X			
CP.1(4)	X			
CP.1(9)			X	
CP.1(10)	X			
CS.1(2)		X		
CS.1(3)		X		
CS.1(4)		X		
CS.1(5)		X		
CS.2(1)		X		
CS.2(2)		X		
CS.2(3)		X		
CS.2(6)			X	
EP.1(2)				X
EP.1(3)			X	
EP.1(4)	X			
EP.1(6)			X	
EP.2(4)			X	
EP.2(5)			X	
EP.2(7)	X			
ES.1(6)			X	
FS.1(1)				X
FS.1(2)	X			
GE.2(2)				X
GE.2(3)				X
GE.2(4)			X	
ID.1(1)	X			
ID.1(3)				X
MO.1(3)	X			
MO.1(4)			X	
MO.1(5)			X	

PP/RD/02

TABLE 4.4-1 (CONT)

SKILL LEVEL REQUIRED TO ANSWER QUESTIONS ON WORKSHEET 4B

QUESTION	ANALYST			
	JUNIOR ANALYST	SENIOR ANALYST	JUNIOR PROGRAMMER	SENIOR PROGRAMMER
MO.1(6)			X	
MO.1(7)			X	
MO.1(8)			X	
MO.1(9)				X
MO.2(5)				X
SD.1(1)	X			
SD.2(1)	X			
SD.2(2)	X			
SD.2(3)	X			
SD.2(4)				X
SD.2(5)			X	
SD.2(6)	X			
SD.2(7)	X			
SD.2(8)	X			
SD.3(1)			X	
SD.3(2)			X	
SD.3(3)			X	
SD.3(4)	X			
SD.3(5)	X			
SD.3(6)			X	
SI.1(2)				X
SI.1(3)				X
SI.1(4)	X			
SI.1(5)	X			
SI.3(1)			X	
SI.4(1)			X	
SI.4(2)	X			
SI.4(3)			X	
SI.4(4)			X	
SI.4(5)				X
SI.4(6)	X			
SI.4(7)			X	
SI.4(8)	X			
SI.4(9)	X			
SI.4(10)	X			
SI.4(11)	X			
SI.4(12)			X	
SI.4(13)			X	
SI.5(1)			X	
SI.5(2)			X	
SI.5(3)				X

TABLE 4.4-1 (CONT)

SKILL LEVEL REQUIRED TO ANSWER QUESTIONS ON WORKSHEET 4B

QUESTION	ANALYST			
	JUNIOR ANALYST	SENIOR ANALYST	JUNIOR PROGRAMMER	SENIOR PROGRAMMER
SI.6(1)		X		
ST.1(1)			X	
ST.1(2)			X	
ST.1(3)			X	
ST.1(4)			X	
ST.1(5)			X	
ST.2(1)			X	
ST.2(2)	X			
ST.2(3)	X			
ST.2(4)	X			
ST.2(5)	X			
ST.3(3)			X	
ST.4(1)	X			
ST.4(2)		X		
ST.4(4)			X	
ST.4(5)			X	
ST.5(1)			X	
ST.5(2)			X	
ST.5(3)			X	
ST.5(4)				X
VS.1(1)	X			
VS.1(2)	X			

PP/RD/04

on our experience, however, we recommend that early worksheets be evaluated by more senior personnel, and that the later worksheets can mostly be evaluated automatically and by more junior programming personnel. The reason for using senior personnel early is to add their experience and knowledge to the interpretation of the system and of metric questions. The decisions made early in the life of a project have great impact on its cost and effectiveness. Later worksheets are of course important, but errors in judgment at that point are less costly to correct than are errors made during requirements analysis.

There are several implications for listing the skill level required to answer each of the questions on each of the worksheets. In general, the greater the skill or experience required to answer a metric question, the more difficult it will be to automate that question. Automation is an important cost reduction tool for the software quality measurement methodology, and can be used in conjunction with understanding of experience needed. In addition, experience is needed to aid in the evaluation of the impact of a "No" or "0" answer to a metric question. This experience can be used to make recommendations as to corrections and effort that should be devoted to a particular problem. The more subjective metric questions should be analyzed by the more senior and experienced staff, in order to best use the talent available. This is particularly true in the earlier worksheets (Worksheets 0, 1, and 2). The later worksheets (3A & 3B, and 4A & 4B) are more easily automated, and tend to contain questions that are more explicit and of a "counting" nature.

Techniques for answering the question. The methods used to answer questions lends itself very easily to establishing some basic catgories. The types of categories we recommend include:

- Counting. Some metric questions may be answered by a straight-forward counting of such things as lines of code, lines of comments, nesting level, and data references. These in general require little decision-making, and may be completed in one pass through a document section or unit of source code.

- Understanding. Some questions require an analyst to read and understand material, in order to decide if the material is clear, complete, logically indented, etc.

RECOMMENDATION — These types of categorization and analysis methods are all reflected in the workbook approach recommended by SAIC. The workbook would be a collection of true worksheets, with each worksheet indicating the categories and information discussed above in this paragraph.

Figure 4.4-1 is a sample of how some of this workbook should look. The workbook would indicate where in particular documents data items are expected to be found. It should also indicate the analyst level of personnel needed to collect and evaluate the data. This would eliminate a great deal of search time. It would also indicate the skill level of the analyst needed to complete each of these new worksheets, and indicate techniques for data collection and evaluation.

The workbook would indicate, for sections of each source document, what questions are to be answered, what data collected, and who can best collect the data. An example of this is the data collected for each "described function." The analyst would examine each function, and note whether it is defined completely, what its name is, and what references to it exist. All this information would be used to answer metric questions on the kind of worksheets currently given in Volume III of the guidebooks. The analyst evaluating the specification would be aware of exactly what data is needed, and where it is to be used on each worksheet.

In addition, the workbook could contain areas that allow the evaluator to add comments and information on any software problem reports generated. Dates of problem report submittal and resolution could also be included.

The workbook approach is also very efficient at gathering data, even using manual methods. The analyst performing the evaluation is not forced into repetitive examination of material, but evaluates it in a meaningful fashion with as few iterations as possible. Another main strength lies in the repeatability of the process. Currently, no way exists to verify a metric score, because no data is retained which supports any conclusions. Using a formal data collection workbook would provide a means of retaining the data, allowing verification, and also allowing the recalculation of scores based on document updates.

These workbooks could be tailored (for reduced sets of factors, criteria, or metrics) just as the current worksheets can be tailored. Data that is not needed for any metric elements would not be collected. To support this tailoring, the workbook should include references to applicable metric elements for each data item to be collected.

4.4.2 Scoring Worksheets 3B and 4B

See: Guidebook Volume III, Appendix A, "Metric Worksheets"

The SQM methodology specifies that one copy of Worksheets 3B and 4B is to be completed for each software unit. A unit is defined in DOD-STD-2168 as the smallest logical entity specified in the detailed design which completely describes a single function in sufficient detail to allow implementing code to be produced and tested independently of other units. The definition also applies to the units as the actual physical entities implemented in the code. In the worksheet context, this definition seems to be what is meant by the term "unit." Measurements are made against each unit, then compiled into CSCI scores.

1. DATA SOURCE: System/Segment Specification, paragraphs 3.1.4.n

METRICS: AC.1(2),AC.1(3),AC.1(4),AC.1(5),CL.1(7),CL.1(8), CP.1(5),CP.1(7) CL.2(2),CP.1(8)

a. List all system functions and defining paragraph on sheet 1 ("Reference" column will be filled in later.)

Sheet 1

Analyst: Pierce

System: ABCD

FUNCTION	DEFINING PARAGRAPH	REFERENCES
1. Monitor COA	3.1.4.1	
2. Analyze COA	3.1.4.2	

Figure 4.4-1 Sample Workbook 0

2. DATA SOURCE: Software Standards and Procedures Manual, paragraph 3.6 or Software Development Plan, paragraph 5.1

METRICS: SI.1(8),SI.1(9),SD.2(1),(Etc.)

Answer each question below, recording where the answer was found (if present) or should have been found (if not present)

QUESTION	ANSWER	LOCATION OF ANSWER
Has a programming standard been established?	Y	SDP 5.1
Has a standard been established for the identification and placements of comments in the unit?	Y	SDP 5.1.1

Figure 4.4-1 (Continued)

b. Copy function list from sheet 1 to sheet 2. For each funct..n, answer the column header and record where answer found or should have been found.

Sheet 2

Analyst:

System:

FUNCTION	DATA TRANSLATION PERFORMED? CL.2(2)	FCT DEFINED? CP.1(7)	ALGORITHMS AND DECISIONS POINTS DESCRIBED CP.1(8)	OUTPUT TRANSMITTED TO OTHER SYSTEM? CL.1(8)	INPUT FROM OTHER SYSTEM? CL.1(7)	ACCURACY REQUIREMENTS BUDGETED HERE? AC.1(2)	INPUTS HAVE [illegible] AN ACCURACY REQ? AC.1(3)	
Monitor COA	Y 2.1.4.16	Y 3.1.4.17	N 3.1.4.18	Y 3.1.4.19	N 3.1.4.19	Y 3.1.4.16	N 3.1.4.1.2	
Analyze COA	N 3.1.4.4.1	N 3.1.4.1.2	. . .					
⋮								

Figure 4.4-1 (Continued)

The advantage of completing a separate Worksheet 3B and 4B for each unit of the code and design lies in unit independence. Worksheet answers may be easily separated and distributed to the authors or evaluators of each software unit. There is a major disadvantage, however, in the bulk of the paper involved.

In all but the smallest systems, the evaluation of Worksheets 3B and 4B, as they currently exist, would create a mountain of paper. As an example, consider the size of the Enemy Course of Action Evaluation Aid (ECOAEA) and the Enemy Sortie Capability Measurement Aid (ESCMA) systems. Each of these systems is small. The ECOAEA consists of 29 files having 150 subroutines or units. The ESCMA consists of 139 programs and procedures.

If each unit were documented in the detailed design, then the 10 pages of Worksheet 3B and 13 pages of Worksheet 4B would be completed for each unit. The result of this would be approximately 10,800 pages just to record metric element answers. On a large system project, this number could quickly become astronomical.

For this reason, we recommend that the SQM methodology be modified. This recommendation concerns only a format change for Worksheets 3B and 4B. If the worksheets were modified to a tabular form, the amount of paper generated could be drastically reduced. In addition, the process of scoring each CSCI would be made easier.

As an example of the reduction in paper bulk, the following tables are proposed for Worksheets 3B and 4B. The modified Worksheet 4B consists of 10 pages, and answers for 15 units may be contained on each page. This means that 333 pages are required to hold answers for 500 modules (500 modules divided by 15 modules per page multiplied by 10 pages per worksheet). Under the current methodology, 6,500 pages (13 pages of Worksheet 4B multiplied by 500 modules) would be required. This is better than a 19:1 reduction. For Worksheet 3B 8 pages were used for the tables. For both worksheets together, the page count for 500 modules is 600 pages using the modified tables, and 11,500 using the SQM method.

Figure 4.4-2 is a sample of these worksheet answer tables.

These modified scoring worksheets can be used with the data collection workbook described in paragraph 4.4.1, above. The workbooks record the original information needed to answer the worksheet questions, but are not themselves directly scorable. These scoring worksheets would be used in conjunction with the workbook in order to provide a place to calculate and record the scores for both Worksheets 3B and 4B.

4.4.3 Metric Worksheets Errors

See: Guidebook Volume III, Appendix A, "Metric Worksheets."

Worksheet 4B

Metric	SD 2(1)	SD 2(2)	SD 2(3)	SD 2(4)	SD 2(5)	SD 2(6)	SD 2(7)	SD 2(8)	SD 3(1)	SD 3(2)	SD 3(3)	SD. 3(4)				SD 3(5)	SD 3(6)	SI 1(2)	SI 1(3)
Unit	Y N N/A	Y N N/A	Y N N/A	Y N N/A	Y N N/A	Y N N/A	Y N N/A	Y N N/A	Y N N/A	Y N N/A	Y N N/A	d	e	f	1- ((e+f) /d)	Y N N/A	Y N N/A	Y N N/A	Y N N/A

FIGURE 4.4-2 SAMPLE ANSWER TABLE FOR WORKSHEET 4B

Some metric questions in the worksheets have typographical and other formatting errors in their text. These questions are identified and described below.

SCORING ERRORS
Many of the questions on Worksheets 3B and 4B are answered by either a "yes" or a "no" response. It is then appropriate for Worksheets 3A and 4A to total the number of "yes" responses. Occasionally, Worksheet 3A or 4A incorrectly asks the analyst to "add applicable unit scores," as shown below:

WORKSHEET 3A or 4A incorrect sample:
a. How many applicable units (score entered on 3B [or 4B])?
b. What is the total score for all applicable units (add applicable unit scores from 3B [or 4B])?
c. Calculate b/a and enter score.

This should be corrected to:
a. How many applicable units (answer of Y or N on 3B [or 4B])?
b. How many units with answer of Y (see 3B [or 4B])?
c. Calculate b/a and enter score.

This correction applies to the following questions:
Worksheet 3A, FS.1(1)

For other Worksheet 3A and 4A questions, the reverse situation occurs. Many of the questions on Worksheets 3B and 4B result in calculated scores between 0 and 1. For some of these, Worksheets 3A or 4A incorrectly ask the analyst to count the number of "units with an answer of Y," as shown below:

Worksheet 3A or 4A sample incorrect version:
a. How many applicable units (answer of Y or N on 3B [or 4B])?
b. How many units with answer of Y (see 3B [or 4B])?
c. Calculate b/a and enter score.

This should be corrected to read:
a. How many applicable units (score entered on 3B [or 4B])?
b. What is total score for all applicable units (add applicable unit scores from 3B [or 4B])?
c. Calculate b/a and enter score.

This correction applies to the following questions:

Worksheet 3A, SI.6(1)
Worksheet 4A, MO.1(4)
Worksheet 4A, SD.3(4)

Worksheet 4A, SI.4(7)
Worksheet 4A, SI.4(8)
Worksheet 4A, SI.4(9)
Worksheet 4A, SI.4(10)
Worksheet 4A, SI.4(11)

Worksheet 3A, SI.6(1)

On Worksheet 3B, this question is:
d. How many unique operations?
e. How many unique operands?
f. How many total operands?
g. Calculate 1 - (2xe)/(dxf) and enter score.

Part (a) should contain "operators", not "operations".

Worksheets 4A and 4B, EP.1(3)

Worksheet 4A contains the question as:
a. How many applicable units (score entered on 4B)?
b. What is total score for all applicable units (add applicable unit scores from 4B)?
c. Calculate b/a and enter score.

Worksheet 4B contains:
d. How many units are required to be optimized for processing efficiency?
e. How many units are optimized for processing efficiency (i.e., compiled using an optimizing compiler or coded in assembly language)?
f. Calculate 1-(e/d) and enter score.

The question should be eliminated entirely from Worksheet 4B.

The Worksheet 4A question should be rewritten. In addition, the calculated score should have been e/d, not 1-(e/d) as shown in Worksheet 4B. Worksheet 4A should then contain:
a. How many units are required to be optimized for processing efficiency?
b. How many units are optimized for processing efficiency (i.e., compiled using an optimizing compiler or coded in assembly language)?
c. Calculate a/b and enter score.

Worksheet 4A, ES.1(7)

The question states:
a. How many total software units?
b. How many software units are optimized for storage efficiency...
c. Calculate 1-(b/a) and enter score.

This results in the score being lower the closer the system matches its requirements. Part (c) should be:
    c. Calculate b/a and enter score.

Worksheet 4A, FS.1(1)

The score "box" for recording the answer to part (c) of this question contains " Y   N   N/A". It should contain room for a numerical answer, followed by "N/A".

Worksheet 4A,4B ID.2(2)

This question does not appear on Worksheet 4B. It states on 4A:

a. How many units in the CSCI?
b. How many units in the CSCI perform external input/output?
c. Calculate 1-(b/a) and enter score.

The question should be redone so that it does appear on 4A as:

a. How many applicable units (answer of Y or N on 4B)?
b. How many units with answer of Y (see 4B)?
c. Calculate 1-(b/a) and enter score.

Worksheet 4B should then contain:

d. Does this unit perform external input/output?

Worksheet 4A,4B ID.2(3)

This question does not appear on Worksheet 4B. It states on 4A:

a. How many units in the CSCI?
b. How many units in the CSCI contain operations dependent on word or character size?
c. Calculate 1-(b/a) and enter score.

The question should be redone so that it does appear on 4A as:

a. How many applicable units (answer of Y or N on 4B)?
b. How many units with answer of Y (see 4B)?
c. Calculate 1-(b/a) and enter score.

Worksheet 4B should then contain:

d. Does this unit contain operations dependent on word or character size?

Worksheet 4A,4B ID.2(4)

This question does not appear on Worksheet 4B. It states on 4A:

a. How many units in the CSCI?
b. How many units in the CSCI contain data element representations tha are machine dependent?
c. Calculate 1-(b/a) and enter score.

The question should be redone so that it does appear on 4A as:

a. How many applicable units (answer of Y or N on 4B)?
b. How many units with answer of Y (see 4B)?
c. Calculate 1-(b/a) and enter score.

Worksheet 4B should then contain:

d. Does this unit contain data element representations that are machine dependent?

Worksheet 4B, AP.3(2)

This question is:

d. How many lines of source code, excluding comments?
e. How many non-HOL lines of code excluding comments?
f. Calculate e/d and enter score.

This means that the more assembly language a routine contains, the higher a score it would get. 100 lines of source code, 30 lines of assembly language is 30/100 or .33 for the score. 100 lines of score code and 90 of them assembly language would get .90 for a score. The reverse relationship should be shown. The question should have part (f) changed to read:

f. Calculate 1-e/d and enter score.

4.4.4 Scoring

See: Guidebook Volume III, Appendix B, "Factor Scoresheets"

There is a typographical error in the scoresheet used for calculating the factor efficiency. Figure 4.4-3 presents the scoresheet as originally shown in the SQM guidebooks. Figure 4.4-4 shows how the scoresheet should be corrected.

4.4.5 Software Quality Evaluation Report

See: Guidebook Volume III, Appendix C, "Software Quality Evaluation Report"

The Data Item Description recommended in the SQM methodology for the Software Quality Evaluation Report includes tables listing metric scores, compiling criteria scores, and compiling factor scores. In addition, the scoresheets used to calculate scores for each factor are to be included.

SAIC recommends eliminating these scoresheets from the reports, and in their place adding one more table to the report. Existing tables already list all scores for metrics, for criteria, and for factors. One more table could be included to list scores for each metric element. The scoresheet data would then be covered by these tables, and would not need to be included at all.

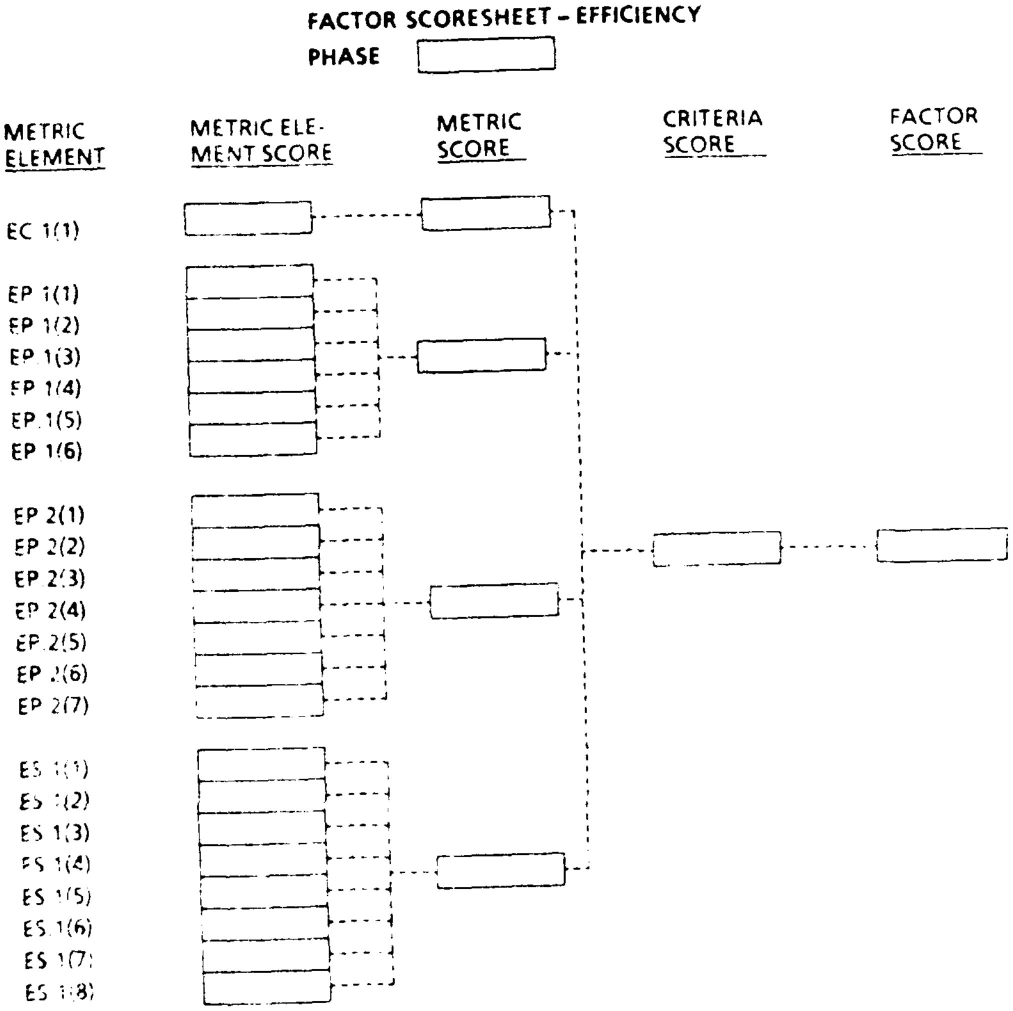


FIGURE 4.4-3 ORIGINAL EFFICIENCY FACTOR SCORESHEET

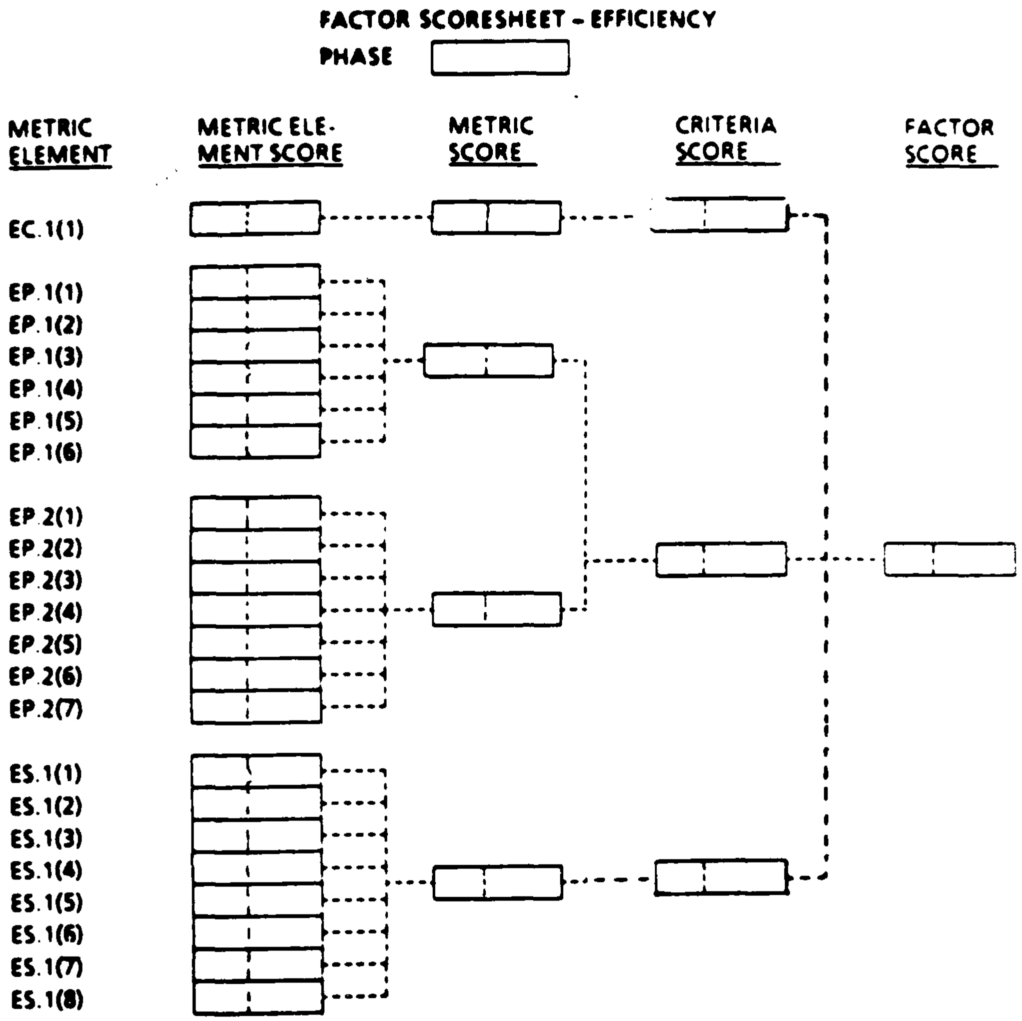


FIGURE 4.4-4 MODIFIED EFFICIENCY FACTOR SCORESHEET

The scoresheets are useful for calculating scores and should be retained, but we recommend that they also be modified. Rather than scoring each factor as is now done, we recommend scoring each criteria. Factor scoresheets would then only show the criteria to be used, and could include the acquisition manager's weighting formula for each criteria used. Even if no other changes are made to the scoresheets, their bulk could be reduced by putting more information on each page. We were able to reduce the amount of paper by three pages by simply cutting and pasting some of the pages together. To accomplish this shortening, we placed the factor INTEGRITY on the same page as the factor EFFICIENCY. VERIFIABILITY and INTEROPERABILITY were both condensed from three pages to two.

We also recommend having a place to put information as needed on each scoresheet. This information could include project, date, analyst, and CSCI being evaluated. Figure 4.4-5 is a sample of the modified factor scoresheet might look. The current scoresheet format would be used in much of its current format, except that data would be included only to the criteria level. Each criteria would then be summed on the factor scoresheet page.

4.4.6 Glossary

See: Guidebook Volume III, Appendix A, "Metric Worksheets"

A glossary needs to be created for each worksheet included in Appendix A of Volume III of the guidebooks. This glossary should contain all terms used in the worksheet, except such words as "the", "calculate", etc. Every software-oriented term should be completely defined and reflect the way that it is used in the particular worksheet. In addition, each term should be used uniquely and consistently for a concept or item. As an example, "data items" and "data references" seem to be used interchangeably on the first two worksheets. Only one term should be used.

4.4.7 Evaluation Procedures

See: Guidebook Volume III, Appendix A, "Metric Worksheets"

Steps to be used as conventions for each question and each situation likely to occur should be defined for each worksheet. These steps would likely be basically the same across the worksheets, with only some specific guidance needed for the separate sheets to answer unique questions.

An example of this guidance is as follows:

1. For multiple part questions that determine the ratio of an occurrence to the possible number of occurrences

[Example: a. Number of calling parameters
b. Number of calling parameters that are control variables.
c. Calculate b/a and enter score.]

If the answer to part (a) is zero, then the score received for part (c) is "N/A".

2. For "all or nothing" questions, the answer must be "N" if any of the measured items do not fully meet the question's statement.

   [Example: Are all inputs documented as to the specific use and limitations of the data?]

   If even 1 input out of 99 is not documented, the answer to the example question is "N". Record the "No" instances so that the acquisition manager and system analysts may have this data available as needed. (The recommended workbook, Section 4.4.1, is a good place for this information to be recorded.)

3. Questions should only be scored as "N/A" under two circumstances. If a question or element has been defined as not applicable for the entire system and eliminated from the scoring process, it may then be scored "N/A". The second circumstance is when these procedures direct such a score (see Procedure #1, above).

4.4.8 Examples

See: Guidebook Volume III, Appendix A, "Metric Worksheets"

While SAIC recommends examples for every step in the methodology, we are particularly interested in seeing examples presented for each metric question in the worksheets. These examples need not be complex, but should illustrate the kind of material that the question concerns, and the focus of its statement. For Worksheets 3A & 3B, and 4A &4B, the examples could be directly related to a small sample of pseudo-language used in design, and actual code used in constructing the equivalent routine or unit, respectively. For other questions, a simple statement of the types of information that would allow a user to determine an answer to the question would be very helpful.

These examples would not be included in each worksheet itself, but in a separate set of material that could be referenced along side of each worksheet during evaluation. The examples could also reference and include the workbooks (see Section 4.4.1) recommended for data collection.

FACTOR SCORESHEET EFFICIENCY

PHASE □

	CRITERIA SCORE		WEIGHTING VALUE		RESULT
[illegible]	□	times	□	=	□
[illegible]	□	times	□	=	□
[illegible] OR	□	times	□		

TOTAL: □

[illegible]

Together with the workbook approach discussed in section 4.4.1, the evaluation procedures discussed in 4.4.7, and the glossary of 4.4.6, nearly any difficulty that an evaluator might have concerning each individual metric question should be easily resolved.

4.4.9 Delete or Move Questions

See: Volume III, Appendix A, "Metric Worksheets"

Some metric questions are asked at inappropriate levels. In general, these questions are of a low level of detail, but occur on worksheets used to analyze requirements or preliminary design documentation. SAIC recommends moving these questions to worksheets 3A & 3B, and 4A & 4B.

4.4.10 All or Nothing Questions

Many of the questions from the worksheets require all or nothing type responses. An example of this is question AM.2(7) on Worksheet 4B: "Is a check performed before processing begins to determine that all data is available?". Other worksheet questions are ratios of possible number of occurrences to the total number of occurrences. While we commented on these all or nothing type questions, we do not recommend changing them at this time. Instead, we recommend that further research be conducted to verify their usefulness, and to validate the statistical validity of combining yes/no questions, all or nothing type questions, and ratio calculation questions.

5.0 REFERENCES

[2167] DOD-STD-2167. Military Standard, Defense System Software Development. 4 June 1985.

[7935] Standard 7935.1-S. Department of Defense, Automated Data Systems Documentation Standards. 13 September 1977.

[BOE-1] Bowen, Thomas P., Wigle, Gary B., and Tsai, Jay T. Specification of Software Quality Attributes. Vol I. RADC-TR-85-37. AD-A153988. February, 1985.

[BOE-2] Bowen, Thomas P., Wigle, Gary B., and Tsai, Jay T. Specification of Software Quality Attributes -- Software Quality Specification Guidebook. Vol II. RADC-TR-85-37. AD-A153989. February, 1985.

[BOE-3] Bowen, Thomas P., Wigle, Gary B., Tsai, Jay T. Specification of Software Quality Attributes -- Software Quality Evaluation Guidebook. Vol III. RADC-TR-85-37. AD-A153990. February, 1985.

[PAR-1] Par Technology. Selected Pages from Senior Battlestaff Decision Aid Development Proposal. 1983.

[PAR-2] Par Technology. Interim Technical Report: Senior Battle Staff Decision Aids, Task I: Planning. Contract # F30602-83-C-0154. 9 March 1984.

[PAR-3] Rome Air Development Center, Statment of Work for Senior Battle Staff Decision Aids. PR No. B-3-3603. 2 Dec 1982.

[PAR-4] Betac Corporation, Enemy Sortie Capability Measurement Aid: Design Plan and Functional Description "Senior Battle Staff Decision Aids". CDRL A003 and A005. Sept 1985.

[PAR-5] Par Technology. Enemy Course of Action Evaluation Aid, Functional Description and Design Plan, Senior Battle Staff Decision Aids. CDRL A003 and A005. July 1984.

[PAR-6] Par Technology. Enemy Course of Action Evaluation Aid Final Functional Description and Design Plan. CDRL A003 and A005. October 1985.

6.0 ACRONYMS

CSC	Computer Software Component
CSCI	Computer Software Configuration Item
DID	Data Item Description
ECOAEA	Enemy Course of Action Evaluation Aid
ESCMA	Enemy Sortie Capability Measurement Aid
MPR	Methodology Problem Report
RADC	Rome Air Development Center
SAIC	Science Applications International Corporation
SQM	Software Quality Measurement
SQMD	Software Quality Measurement Demonstration
TPR	Technical Problem Report
WS	Worksheet

APPENDIX A

SURVEY QUESTIONNAIRE

SOFTWARE QUALITY MEASUREMENT DEMONSTRATION PROJECT

As part of the evaluation of the Specification of Software Quality Attributes methodology, SAIC is following the procedures outlined to measure the software quality of two decision aid systems. These programs are the Enemy Sortie Capability Measurement Aid, and the Enemy Course of Action Evaluation Aid. Both aids are part of the Senior Battlestaff Decision Aid project.

Because of your involvement and experience with these aids, we are asking you to fill out some forms and return them to us. These forms will be used to determine what quality goals should be specified for each major function of each decision aid. They will also be used to evaluate the adequacy of this method of quality goal specification.

In addition to the goal specification forms, we will ask you to fill out a form giving us your response to this method of goal specification. Please keep track of the time you spend on this complete task so that you may completely fill in the last feedback form.

Figure 1 presents the 13 software quality factors we are concerned with, along with definitions of each. To aid in understanding, these factors are grouped under major concerns (performance of the system, how it is to be designed, and how adaptable it needs to be).

Tables 1 and 2 are quality survey forms we would like you to fill out, one for each decision aid. If you are not familiar with both aids, fill in whichever form is appropriate. For each decision aid function listed on the left, please rate the quality goal you feel is desirable for each quality factor. If you can, please provide us with two ratings. The first rating is the goal you believe you would have specified had you filled out this form before [illegible] aid [illegible] development. The second rating is the goal you now believe to be important. If you cannot provide the first rating, just fill in your best current estimate.

Use "E" [illegible] indicate that [illegible] "E" for excellent, "G" for good, "A" for average, and "N" for not applicable or not important. Please fill in some [illegible] for each function and each factor.

Table 3 presents the definitions of the [illegible] software oriented criteria. Factors are made up of [illegible] which are further [illegible] into metrics which are the actual [illegible] elements measured to determine system quality.

To aid us in our methodology evaluation, please fill out Tables 4 and 5, rating the criteria against each function as you did the factors. These ratings will help us weight the criteria's importance within each factor.

The last form we will ask you to complete is Figure 2. This form gives us information about you and your reactions to this survey.

ACQUISITION CONCERN	QUALITY FACTOR	DEFINITION
PERFORMANCE	EFFICIENCY	RELATIVE EXTENT TO WHICH A RESOURCE IS UTILIZED (i.e., STORAGE SPACE, PROCESSING TIME, COMMUNICATION TIME)
	INTEGRITY	EXTENT TO WHICH THE SOFTWARE WILL PERFORM WITHOUT FAILURES DUE TO UNAUTHORIZED ACCESS TO THE CODE OR DATA WITHIN A SPECIFIED TIME PERIOD
	RELIABILITY	EXTENT TO WHICH THE SOFTWARE WILL PERFORM WITHOUT ANY FAILURES WITHIN A SPECIFIED TIME PERIOD
	SURVIVABILITY	EXTENT TO WHICH THE SOFTWARE WILL PERFORM AND SUPPORT CRITICAL FUNCTIONS WITHOUT FAILURES WITHIN A SPECIFIED TIME PERIOD WHEN A PORTION OF THE SYSTEM IS INOPERABLE.
	USABILITY	RELATIVE EFFORT FOR TRAINING OR SOFTWARE OPERATION (e.g., FAMILIARIZATION, INPUT PREPARATION, EXECUTION, OUTPUT INTERPRETATION)
DESIGN	CORRECTNESS	EXTENT TO WHICH THE SOFTWARE CONFORMS TO ITS SPECIFICATIONS AND STANDARDS.
	MAINTAINABILITY	EASE OF EFFORT FOR LOCATING AND FIXING A SOFTWARE FAILURE WITHIN A SPECIFIED TIME PERIOD.
	VERIFIABILITY	RELATIVE EFFORT TO VERIFY THE SPECIFIED SOFTWARE OPERATION AND PERFORMANCE
ADAPTATION	EXPANDABILITY	RELATIVE EFFORT TO INCREASE THE SOFTWARE CAPABILITY OR PERFORMANCE BY ENHANCING CURRENT FUNCTIONS OR BY ADDING NEW FUNCTIONS OR DATA.
	FLEXIBILITY	EASE OF EFFORT FOR CHANGING THE SOFTWARE MISSIONS, FUNCTIONS, OR DATA TO SATISFY OTHER REQUIREMENTS.
	INTEROPERABILITY	RELATIVE EFFORT TO COUPLE THE SOFTWARE OF ONE SYSTEM TO THE SOFTWARE OF ANOTHER SYSTEM.
	PORTABILITY	RELATIVE EFFORT TO TRANSPORT THE SOFTWARE FOR USE IN ANOTHER ENVIRONMENT (HARDWARE CONFIGURATION AND/OR SOFTWARE SYSTEM ENVIRONMENT).
	REUSABILITY	RELATIVE EFFORT TO CONVERT A SOFTWARE COMPONENT FOR USE IN ANOTHER APPLICATION.

Figure 1. Quality Factors

Table 1. ESCMA Quality Goals

Enemy Sortie Capability Measurement Aid Software Function	Efficiency	Integrity	Reliability	Survivability	Usability	Correctness	Maintainability	Verifiability	Expandability	Flexibility	Interoperability	Portability	Reusability

Table 2. ECOAEA Quality Goals

Enemy Course of Action Evaluation Aid	Efficiency	Integrity	Reliability	Survivability	Usability	Correctness	Maintainability	Verifiability	Expandability	Flexibility	Interoperability	Portability	Reusability

ACQUISITION CONCERN	CRITERION	DEFINITION
PERFORMANCE	ACCURACY	THOSE CHARACTERISITICS OF SOFTWARE WHICH PROVIDE THE REQUIRED PRECISION IN CALCULATIONS AND OUTPUTS
	ANOMALY MANAGEMENT	THOSE CHARACTERISITCS OF SOFTWARE WHICH PROVIDE FOR CONTINUITY OF OPERATIONS UNDER AND RECOVERY FROM NON NOMINAL CONDITIONS.
	AUTONOMY	THOSE CHARACTERISTICS OF SOFTWARE WHICH DETERMINE ITS NON-DEPENDENCY ON INTERFACES AND FUNCTIONS.
	DISTRIBUTEDNESS	THOSE CHARACTERISTICS OF SOFTWARE WHICH DETERMINE THE DEGREE TO WHICH SOFTWARE FUNCTIONS ARE GEOGRAPHICALLY OR LOGICALLY SEPARATED WITHIN THE SYSTEM.
	EFFECTIVENESS-COMM	THOSE CHARACTERISTICS OF THE SOFTWARE WHICH PROVIDE FOR MINIMUM UTILIZATION OF COMMUNICATIONS RESOURCES IN PERFORMING FUNCTIONS.
	EFFECTIVENESS-PROCESSING	THOSE CHARACTERISTICS OF THE SOFTWARE WHICH PROVIDE FOR MINIMUM UTILIZATION OF PROCESSING RESOURCES IN PERFORMING FUNCTIONS.
	EFFECTIVENESS-STORAGE	THOSE CHARACTERISTICS OF THE SOFTWARE WHICH PROVIDE FOR MINIMUM UTILIZATION OF STORAGE RESOURCES.
	OPERABILITY	THOSE CHARACTERISITCS OF SOFTWARE WHICH DETERMINE OPERATIONS AND PROCEDURES CONCERNED WITH OPERATION OF SOFTWARE AND WHICH PROVIDE USEFUL INPUTS AND OUTPUTS WHICH CAN BE ASSIMILATED.
	RECONFIGURABILITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE FOR CONTINUITY OF SYSTEM OPERATION WHEN ONE OR MORE PROCESSORS, STORAGE UNITS, OR COMMUNICATION LINKS FAILS.
	SYSTEM ACCESSIBILITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE FOR CONTROL AND AUDIT OF ACCESS TO THE SOFTWARE AND DATA.
	TRAINING	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE TRANSITION FROM CURRENT OPERATION AND PROVIDE INITIAL FAMILIARIZATION.
DESIGN	COMPLETENESS	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE FULL IMPLEMENTATION OF THE FUNCTIONS REQUIRED.
	CONSISTENCY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE FOR UNIFORM DESIGN AND IMPLEMENTATION TO THE REQUIREMENTS WITH RESPECT TO THE SPECIFIED DEVELOPMENT
	TRACEABILITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE A THREAD OF ORIGIN FROM THE IMPLEMENTATION TO THE REQUIREMENTS WITH RESPECT TO THE SPECIFIED DEVELOPMENT ENVELOPE AND OPERATIONAL ENVIRONMENT.
	VISIBILITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE STATUS MONITORING OF THE DEVELOPMENT AND OPERATION.
ADAPTATION	APPLICATION INDEPENDENCE	THOSE CHARACTERISTICS OF SOFTWARE WHICH DETERMINE ITS NONDEPENDENCY ON DATABASE SYSTEM MICROCODE, COMPUTER ARCHITECTURE, AND ALGORITHMS.
	AUGMENTABILITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE FOR UNIFORM DESIGN AND IMPLEMENTATION TECHNIQUES AND NOTATION.
	COMMONALITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE FOR THE USE OF INTERFACE STANDARDS FOR PROTOCOLS, ROUTINES, AND DATA REPRESENTATIONS.
	DOCUMENT ACCESSIBILITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE FOR EASY ACCESS TO SOFTWARE AND SELECTIVE USE OF ITS COMPONENTS.
	FUNCTIONAL OVERLAP	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE COMMON FUNCTIONS TO BOTH SYSTEMS.
	FUNCTIONAL SCOPE	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE COMMONALITY OF FUNCTIONS AMONG APPLICATIONS.
	GENERALITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE BREADTH TO THE FUNCTIONS PERFORMED WITH RESPECT TO THE APPLICATION.
	INDEPENDENCE	THOSE CHARACTERISTICS OF SOFTWARE WHICH DETERMINE ITS NON-DEPENDENCY ON SOFTWARE ENVIRONMENT (COMPUTING SYSTEM, OPERATING SYSTEM UTILITIES, INPUT, OUTPUT ROUTINES, LIBRARIES).
	SYSTEM CLARITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE FOR CLEAR DESCRIPTION OF PROGRAM STRUCTURE IN A NON-COMPLEX AND UNDERSTANDABLE MANNER.
	SYSTEM COMPATIBILITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE THE HARDWARE, SOFTWARE, AND COMMUNICATION COMPATIBILITY OF TWO SYSTEMS.
	VIRTUALITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PRESENT A SYSTEM THAT DOES NOT REQUIRE USER KNOWLEDGE OF THE PHYSICAL, LOGICAL, OR TOPOLOGICAL CHARACTERISTICS
GENERAL	MODULARITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE A STRUCTURE OF HIGHLY COHESIVE COMPONENTS WITH OPTIMUM COUPLING.
	SELF-DESCRIPTIVENESS	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE EXPLANATION OF THE IMPLEMENTATION OF FUNCTIONS.
	SIMPLICITY	THOSE CHARACTERISTICS OF SOFTWARE WHICH PROVIDE FOR DEFINITION AND IMPLEMENTATION OF FUNCTIONS IN THE MOST NONCOMPLEX AND UNDERSTANDABLE MANNER.

Table 3. Software-Oriented Criteria Definitions

Table 4. ESCMA Criteria Ratings

ESCMA FUNCTIONS

FACTOR AND ASSOCIATED CRITERIA														
Efficiency: Effectiveness of Communication														
Efficiency: Effectiveness of Processing														
Efficiency: Effectiveness of Storage														
Integrity: System Accessibility														
Reliability: Anomaly Management														
Reliability: Accuracy														
Reliability: Simplicity														
Survivability: Anomaly Management														
Survivability: Autonomy														
Survivability: Distributedness														
Survivability: Modularity														
Usability: Operability														
Usability Training														

Table 4. ESCMA Criteria Ratings (Continued)

ESCMA FUNCTIONS

FACTOR AND ASSOCIATED CRITERIA														
Correctness: Completeness														
Correctness: Consistency														
Correctness: Traceability														
Maintainability: Consistency														
Maintainability: Visibility														
Maintainability: Modularity														
Maintainability: Self-Descriptiveness														
Maintainability: Simplicity														
Verifiability: Visibility														
Verifiability: Modularity														
Verifiability: Self-Descriptiveness														
Verifiability Simplicity														
Expandability Augmentability														

Table 4. ESCMF Criteria Ratings (Continued)

ESCMF FUNCTIONS

FACTOR AND ASSOCIATED CRITERIA														
Expandability Generality														
Expandability Virtuality														
Expandability Modularity														
Expandability Self Descriptiveness														
Expandability Simplicity														
Flexibility Generality														
Flexibility Modularity														
Flexibility Self Descriptiveness														
Flexibility Simplicity														
Interoperability Commonality														
Interoperability Functional Development														
Interoperability Independence														
Interoperability System Compatibility														

Table 4. ESCMA Criteria Ratings (Continued)

ESCMA FUNCTIONS

FACTOR AND ASSOCIATED CRITERIA														
Interoperability: Modularity														
Portability: Independence														
Portability: Modularity														
Portability: Self-Descriptiveness														
Reusability: Application Independence														
Reusability: Document Accessibility														
Reusability: Functional Scope														
Reusability: Generality														
Reusability: Independence														
Reusability: System Clarity														
Reusability: Modularity														
Reusability: Self Descriptiveness														
Reusability: Simplicity														

Table 5. ECOAEA Criteria Ratings

ECOAEA FUNCTIONS

FACTOR AND ASSOCIATED CRITERIA														
Efficiency Effectiveness of Communication														
Efficiency Effectiveness of Processing														
Efficiency Effectiveness of Storage														
Integrity System Accessibility														
Reliability Anomaly Management														
Reliability Accuracy														
Reliability Simplicity														
Survivability Anomaly Management														
Survivability Autonomy														
Survivability Distributedness														
Survivability Modularity														
Usability Operability														
Usability Training														

Table 5. ECOAEA Criteria Ratings (Continued)

ECOAEA FUNCTIONS

FACTOR AND ASSOCIATED CRITERIA														
Correctness: Completeness														
Correctness: Consistency														
Correctness: Traceability														
Maintainability: Consistency														
Maintainability: Visibility														
Maintainability: Modularity														
Maintainability: Self-Descriptiveness														
Maintainability: Simplicity														
Verifiability: Visibility														
Verifiability: Modularity														
Verifiability: Self Descriptiveness														
Verifiability: Simplicity														
Expandability: Augmentability														

Table 5. ECOAEA Criteria Ratings (Continued)

FACTOR AND ASSOCIATED CRITERIA	ECOAEA FUNCTIONS													
Expandability: Generality														
Expandability: Virtuality														
Expandability: Modularity														
Expandability: Self-Descriptiveness														
Expandability: Simplicity														
Flexibility: Generality														
Flexibility: Modularity														
Flexibility: Self-Descriptiveness														
Flexibility: Simplicity														
Interoperability: Commonality														
Interoperability: Functional Development														
Interoperability: Independence														
Interoperability: System Compatibility														

Table 5. ECOAEA Criteria Rating (continued)

ECOAEA FUNCTIONS

FACTOR AND ASSOCIATED CRITERIA														
Interoperability: Modularity														
Portability: Independence														
Portability: Modularity														
Portability: Self Descriptiveness														
Reusability: Application Independence														
Reusability: Document Accessibility														
Reusability: Functional Scope														
Reusability: Generality														
Reusability: Independence														
Reusability: System Clarity														
Reusability: Modularity														
Reusability: Self Descriptiveness														
Reusability: Simplicity														

SURVEY RESPONSE

Respondent: ______________________

Date: ______________________

1. What is your role in the decision aid project? ______________________

____

2. How much time did you spend on this survey? ______________________

   On factor goal rating? ______________________

3. Are the quality factors meaningful to you? ______________________

____

4. Are there other aspects of quality you would have identified? If so, what are they? __________

____

____

5. Were the instructions on how to specify the qualities clear? If not, which ones were unclear? __________

____

____

6. Are quantitative guidelines needed (e.g., what reliability means in terms of a number like .95)? __________

____

____

7. Would guidelines that state industry averages help? ______________________

____

8. Do you feel both factor and criteria ratings should be included in this packet? __________

____

9. Other comments: ______________________

____

____

____

____

Figure 2. Survey Feedback

Appendix B -- Criterion Labor Effort Figures

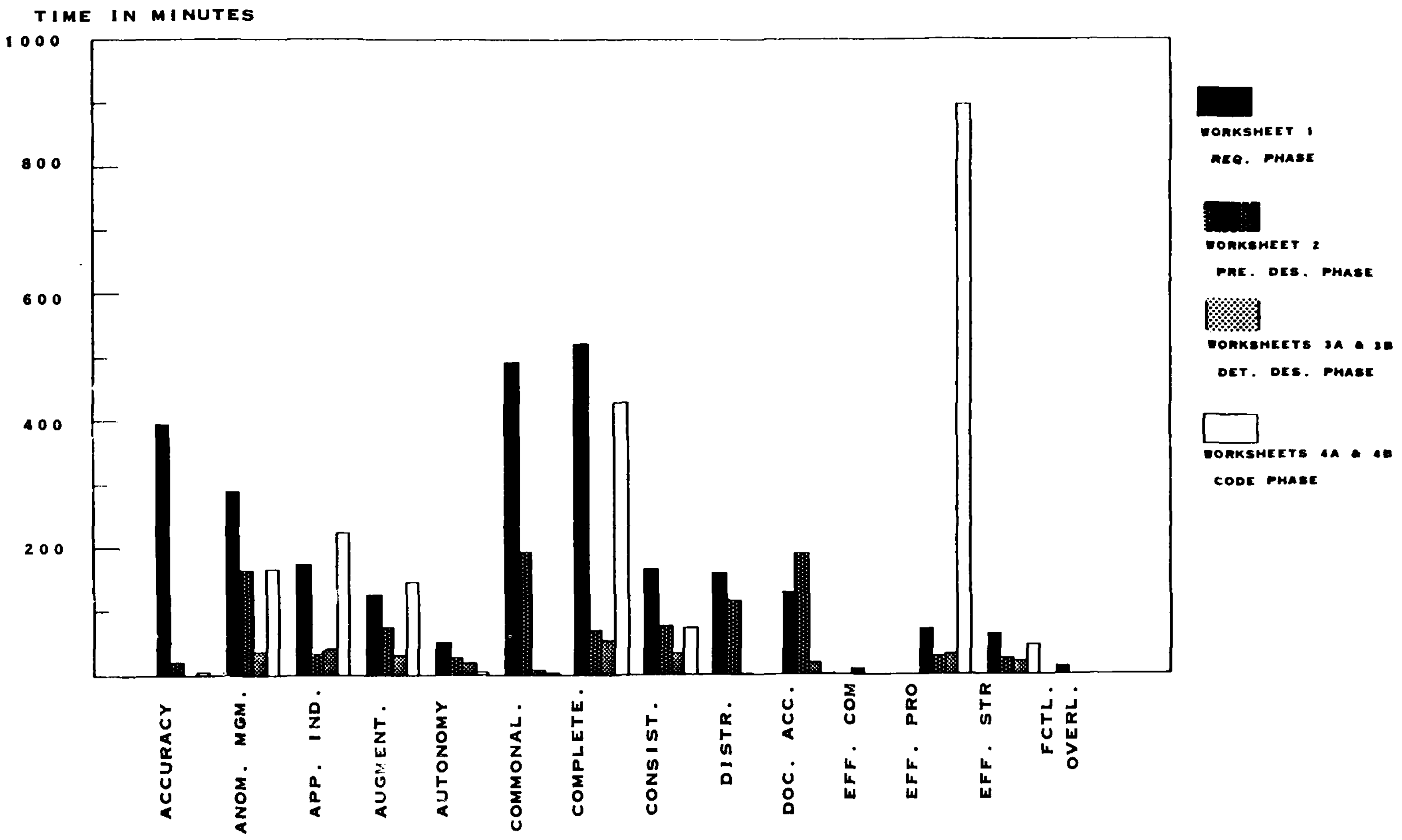


Figure B-1. Criteria Evaluation Time for ESCMA

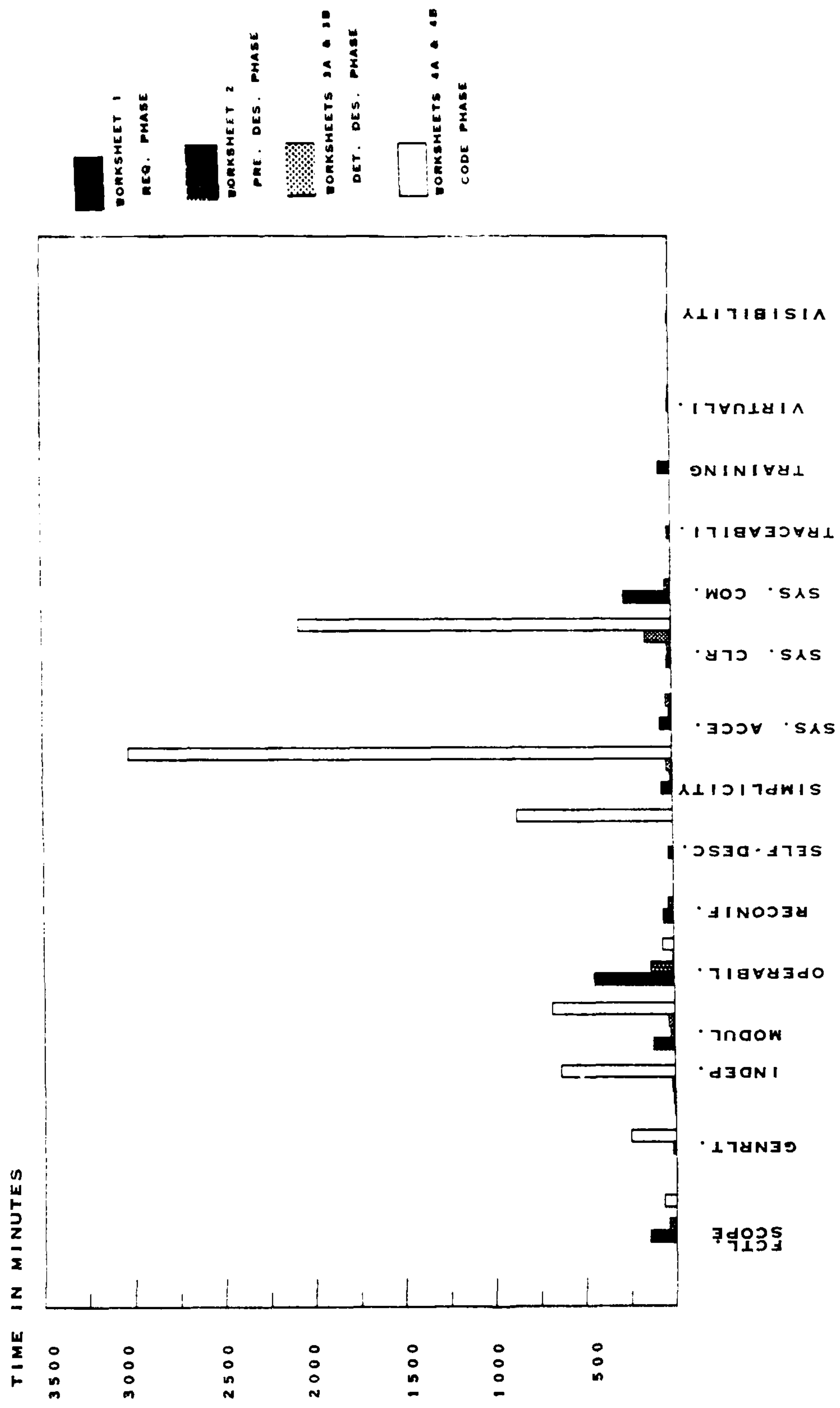


Figure B-1 (Continued). Criteria Evaluation Time for ESCMA

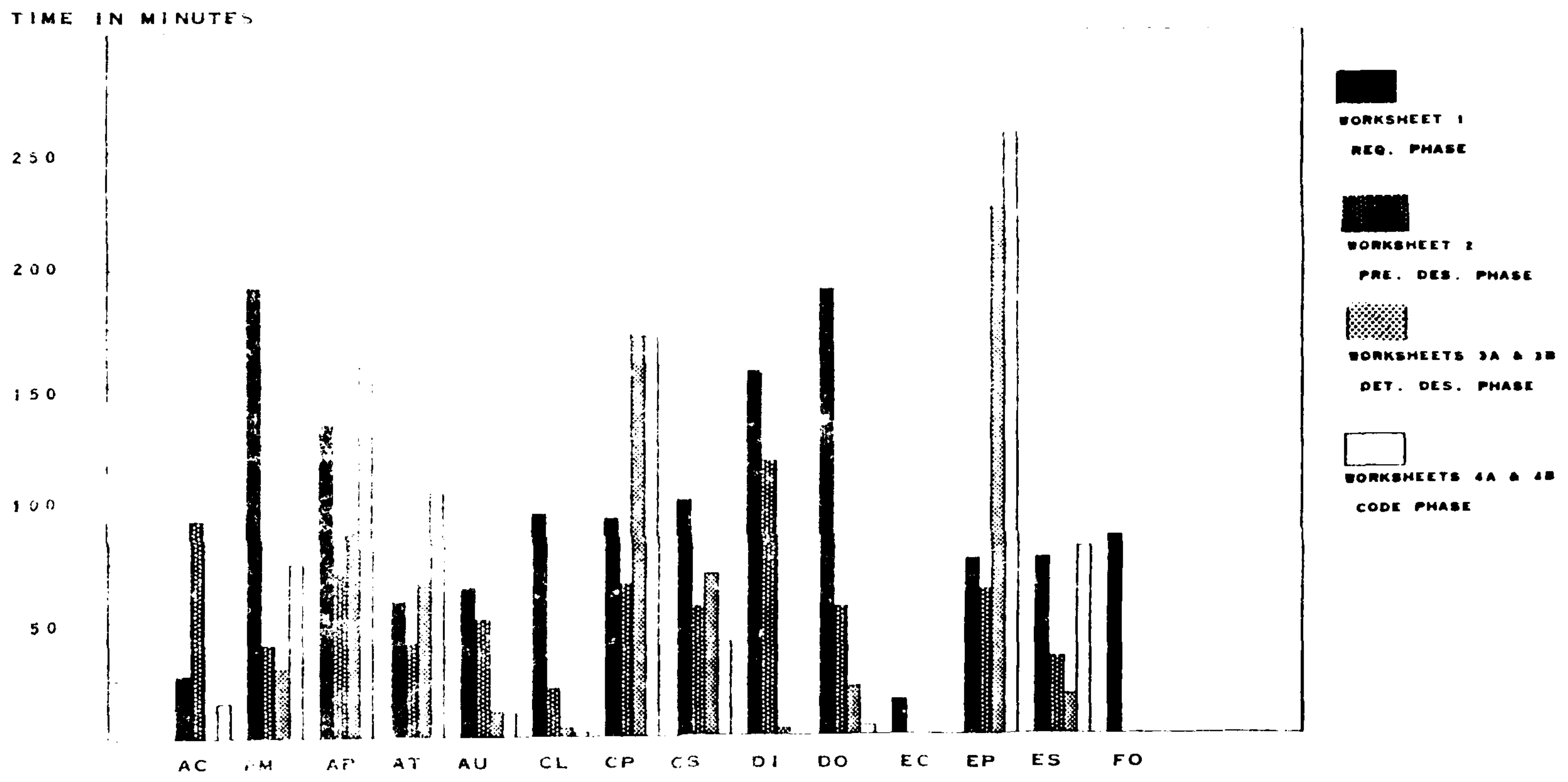


Figure B-2. Criteria Evaluation Time for ECOAEA

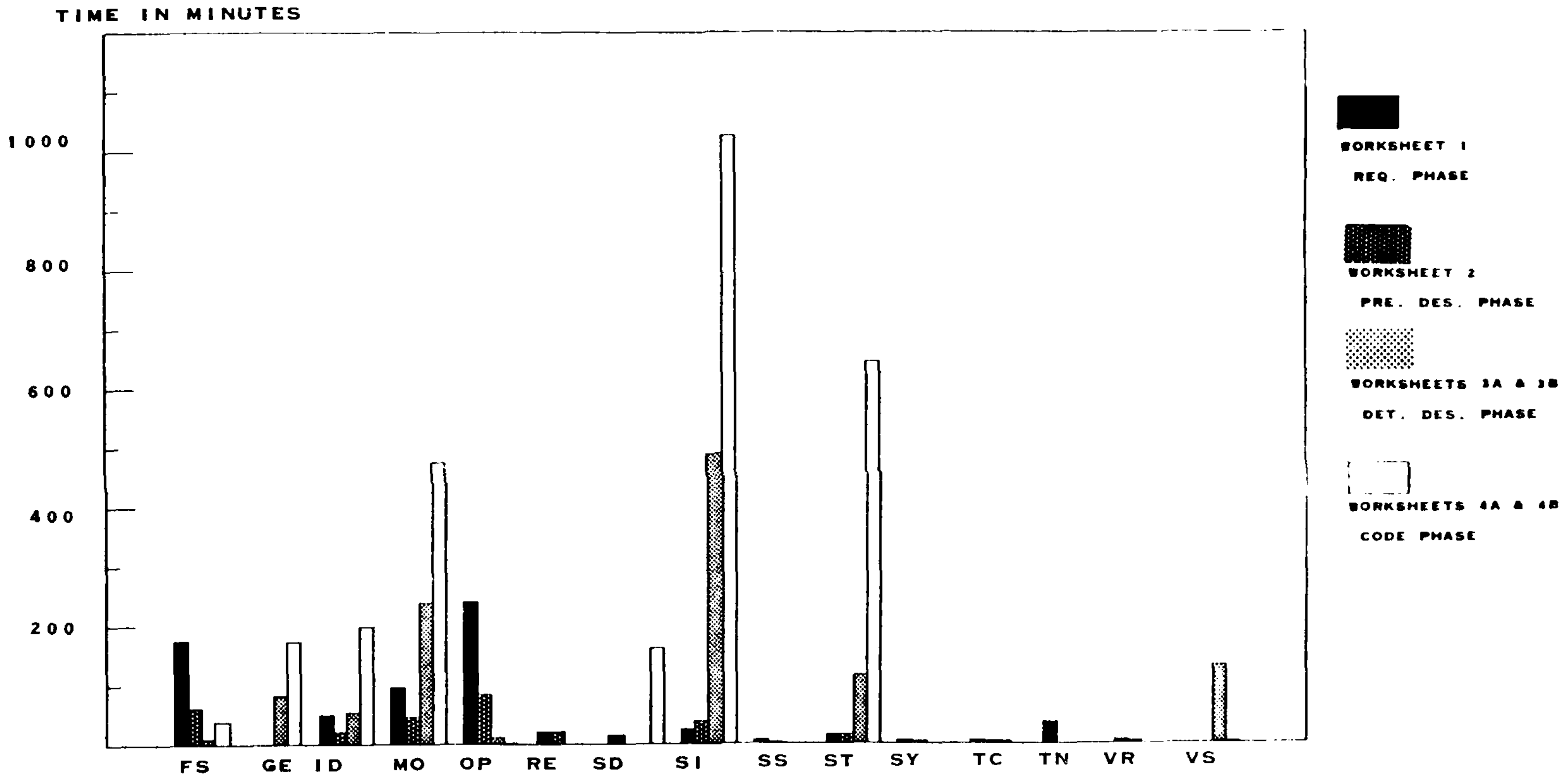


Figure B-2 (Continued). Criteria Evaluation Time for ECOAEA

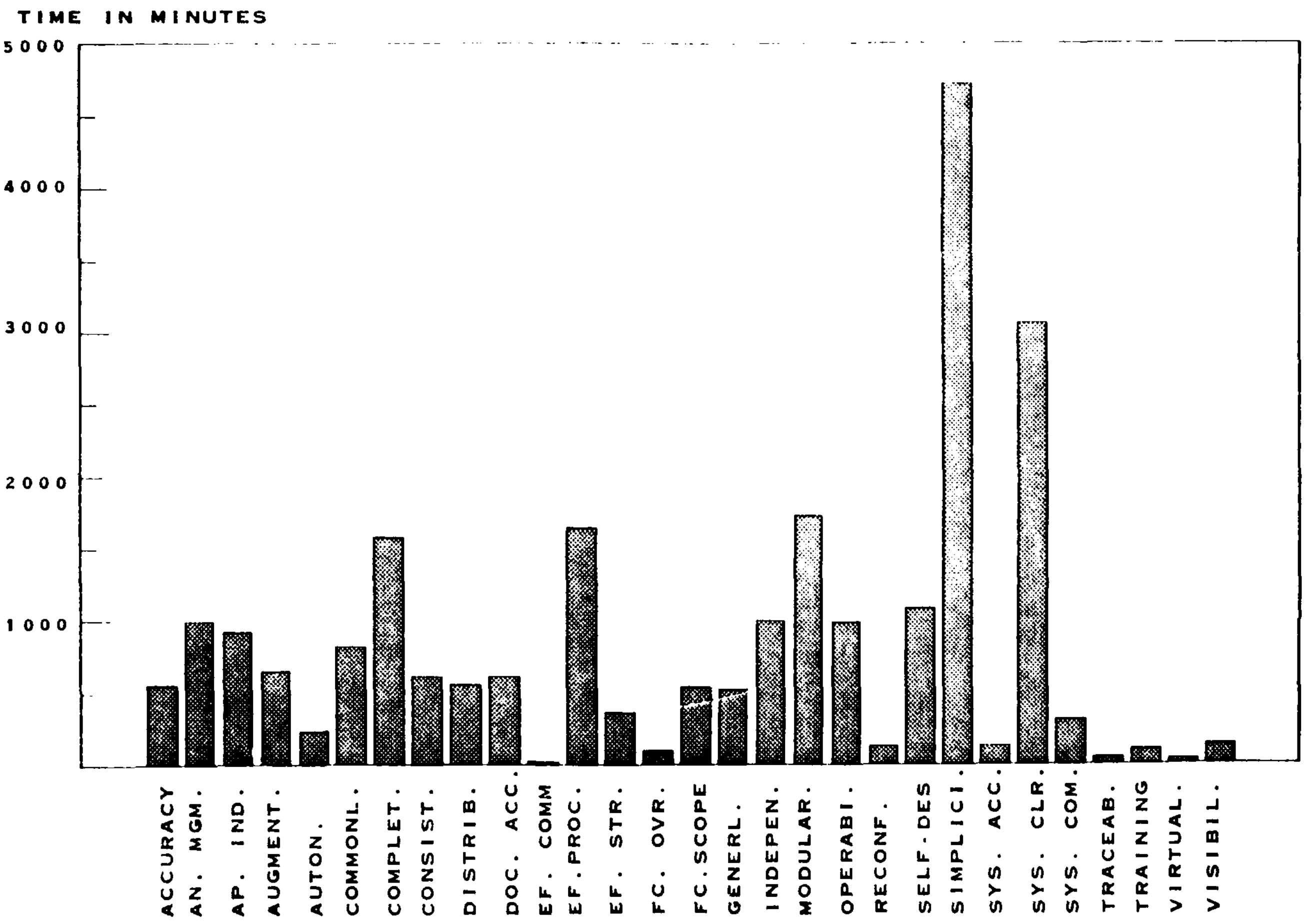


Figure B-3. Criteria Evaluation Time for Both Decision Aids

END
DATE
FILMED
6-1988
DTIC